PC-8001

マシン語活用ハンドブック

初級編─内部サブルーチンのすべて

秀和システムトレーディング株式会社

PC-8001
マシン語活用ハンドブック

―――初級編―内部サブルーチンのすべて―――

秀和システムトレーディング株式会社

まえがき

今日，パーソナルコンピュータと云えばBASICと言われる程，BASICはパーソナルコンピュータの普及に多大な役割を演じています．BASICは，プログラミングが容易，ソース・プログラムの修正が簡単，入力したプログラムをすぐに実行出来るなど優れた特長を有しています．しかし一方，演算速度が遅い，メモリの利用効率があまり良くない，入出力命令の拡張性がやや乏しい等，インタプリタ形式であるがゆえの短所も併せ持っています．

これらの短所を克服してパーソナルコンピュータを更に高度に活用するにはどうしたら良いか？

この問に対する一般的な解答として「それは機械語を用いることで解決する.」と言われています．

では，パーソナルコンピュータにおいて機械語を活用するには誰に相談すれば良いか，どんな本を読めば良いか？

本書は，PC-8001ユーザーのこの様な疑念に応えるために，私達が独自にN-BASIC内部サブルーチンの解析を行った結果を，ここに発表するものであります．内部サブルーチンの使い方，IOCSの仕様，インタプリタの機械語レベルでの応用例，更にインタプリタの構造についてのフローチャートを用いた説明にも触れ，機械語サブルーチンを手軽に応用出来る様に配慮いたしました．

本書が読者のパーソナルコンピュータ活用，研究の一助となり，更に高度な利用技術が発揮されることを願ってやみません．

なお，本書の執筆，出版にあたりお世話になりました石井晴正氏，橋本正樹氏，小牧自行氏，新家弘健氏，前川田井介氏，本間実氏，吉村博文氏，今井勇吉氏，納谷佳子嬢に深く感謝いたします．

1982年6月

FORESIGHT企画部代表　　川 村　 清

目　次

1
N-BASIC インタプリタ解析資料集

この章は，２４KバイトのN―BASICに含まれている多くの有用なシステム・サブルーチンの中から，一般ユーザーの利用頻度が高いと思われるものを選んで，資料集としてまとめたものです．

なお，本章にまとめた解析内容だけでは間に合わず，より高度な応用を考えておられる方々には，御自分でN―BASICを解析なさる事を勧めます．最高傑作であるN―BASICは，その内部を理解することによって，さらに高度な応用に応じてくれる事でしょう．N―BASICの明細につきましては，本書と同時に発刊いたします,「PC―8001 BASIC SOURCE PROGRAM LISTINGS」を参照していただきたいと思います．マイクロソフト・ベーシック解析のお役に立つことでしょう。

本章の各ルーチンは，エントリ・アドレスによってソーティングされ，分類されています．

▶アドレス　各ルーチンのエントリ・ポイントを示します．

名称　各ルーチンの目的を簡単に示します．

レジスタ　各ルーチンの実行後，内容が変更されるレジスタを示します．したがって，これらのレジスタの内容を保存しなければならない場合には，ユーザー・プログラム中でのレジスタ退避が必要です．なお，N―BASICインタプリタでは，アルタネーティブ・レジスタおよびインデクス・レジスタは一切使用していません．

内容　各ルーチンの詳しい使用方法，入出力パラメータなどを説明した上で場合によっては，関連した予備知識も示します．

例　各ルーチンを利用した短かいサブルーチンの例を，ザイログ形式のニーモニックで示します．

フロー・チャート　主要と思われるルーチンにはフロー・チャートをつけておきました．解析，御利用の際にお役立てください．

なお，システム・サブルーチンの他にも，主要なデータ・エリア等を加えておきました．N―BASICインタプリタ内には，これらの他にも１～２バイト程度のデータ・エリアが数多く存在します．逆アセンブルを行なう場合などには十分御注意ください．

▶アドレス　０００0Ｈ

名称　ＰＣ－８００１リセット

レジスタ　――――

内容　Ｎ―ＢＡＳＩＣのイニシアライズを行ない，ストップ・キーの入力があれば，００６ＡＨ番地のホット・スタートに，押されていなければ，１７５７Ｈ番地にジャンプしてコールド・スタートを行ないます．
イニシアライズ中は，ＦＦＦＥＨ番地以下をスタックとして使用します．

▶アドレス　０００８Ｈ

名称　Ｎ－ＢＡＳＩＣホット・スタート

レジスタ　――――

Ｎ―ＢＡＳＩＣのホット・スタートである００６ＡＨ番地へジャンプします．

▶アドレス　００１０Ｈ

名称　Ｎ－ＢＡＳＩＣテキスト解釈

レジスタ　Ａ，Ｆ，Ｂ，Ｃ，Ｄ，Ｅ，Ｈ，Ｌ

内容　４２５９Ｈ番地からのテキスト解析ルーチンにジャンプします．
詳しくは，４２５９Ｈ番地のフロー・チャートを参照してください．

▶アドレス　００１３Ｈ

名称　N－BASICホット・スタート

レジスタ　―――――

内容　N－BASICのホット・スタートである006AH番地へジャンプします.

▶アドレス　0018H

名称　周辺機器への1バイト出力

レジスタ　全て保存

内容　40A6H番地即ち周辺機器への1バイト出力サブルーチンにジャンプします.
RST　18Hによってコールする事ができます.
詳しくは，40A6H番地を参照してください.

▶アドレス　002BH

名称　プリンタへの1バイト出力

レジスタ　全て保存

内容　プリンタへの1バイト出力を行なう，0D60H番地にジャンプします.
詳しくは，0D60H番地を参照してください.

▶アドレス　006AH

名称　N－BASICホット・スタート

レジスタ　――――

内容　N－BASICの初期設定を行ない，ホット・スタートを行ないます．
画面クリアやタイマの初期設定などは行ないますが，N－BASICのテキスト・プログラムは保存されています．
通常，N－BASICに制御を移す場合には，画面などのクリアを行なわずに，コマンド待ちを行う，0081H番地へジャンプした方が良いでしょう．

▶アドレス　0081H

名称　N－BASICコマンド待ち

レジスタ　――――

内容　イニシアライズなどを全く行なわずに，プロンプト「Ok」をCRT画面に表示し，N－BASICのスクリーン・エディット・モードに入ります．
通常，このアドレスにジャンプする事によって，N－BASICに制御を移す事ができます．
又，0081H番地から，F1B0番地へのフックを行なっています．

▶アドレス　0099H

名称　ディスク入出力

レジスタ　F

内容　ディスク・ユニットへの入出力を行なうためのサブルーチンで，マシン語のプログラム中から呼びだす事によってディスク・ユニットへの高速な入出力を行なう事ができます．

以下に，入力パラメータをあげておきますが，詳しくは本書の本文中に述べられています．

Aレジスタ　←—入出力実行セクタ数
Bレジスタ　←—トラック・ナンバ
Cレジスタ　←—セクタ・ナンバ
DEレジスタ対←—入出力開始アドレス

EDC5H番地←—0
EDC6H番地←—ドライブ・ナンバ（0～3）

フラグ（F）　←—入出力の区別
CY←—1：　ライト
Z　←—0：　リード
Z　←—1：　ベリファイ

▶アドレス　00CBH

名称　拡張システム・チェック

レジスタ　A，F

内容　ディスク・ユニットなどの拡張システムが接続されていなければ，ゼロ・フラグ（Z）を1にセットします．
接続の有無は，入力ポート40H番地の第3ビットを調べて決定します．

▶アドレス　00D2H

名称　ディスク初期化

レジスタ　A

内容　コントローラを初期化し，ディスクが何台つながっているかを調べ，Aレジスタに入れてもどります.

Ｄｉｓｋ－ＢＡＳＩＣ使用時にホット・リセットをかけてしまった場合には，モニタから，この番地にジャンプする事によって，Ｄｉｓｋ－ＢＡＳＩＣを起動し直す事ができます.

▶アドレス　０２５７Ｈ

名称　１文字表示

レジスタ　全て保存

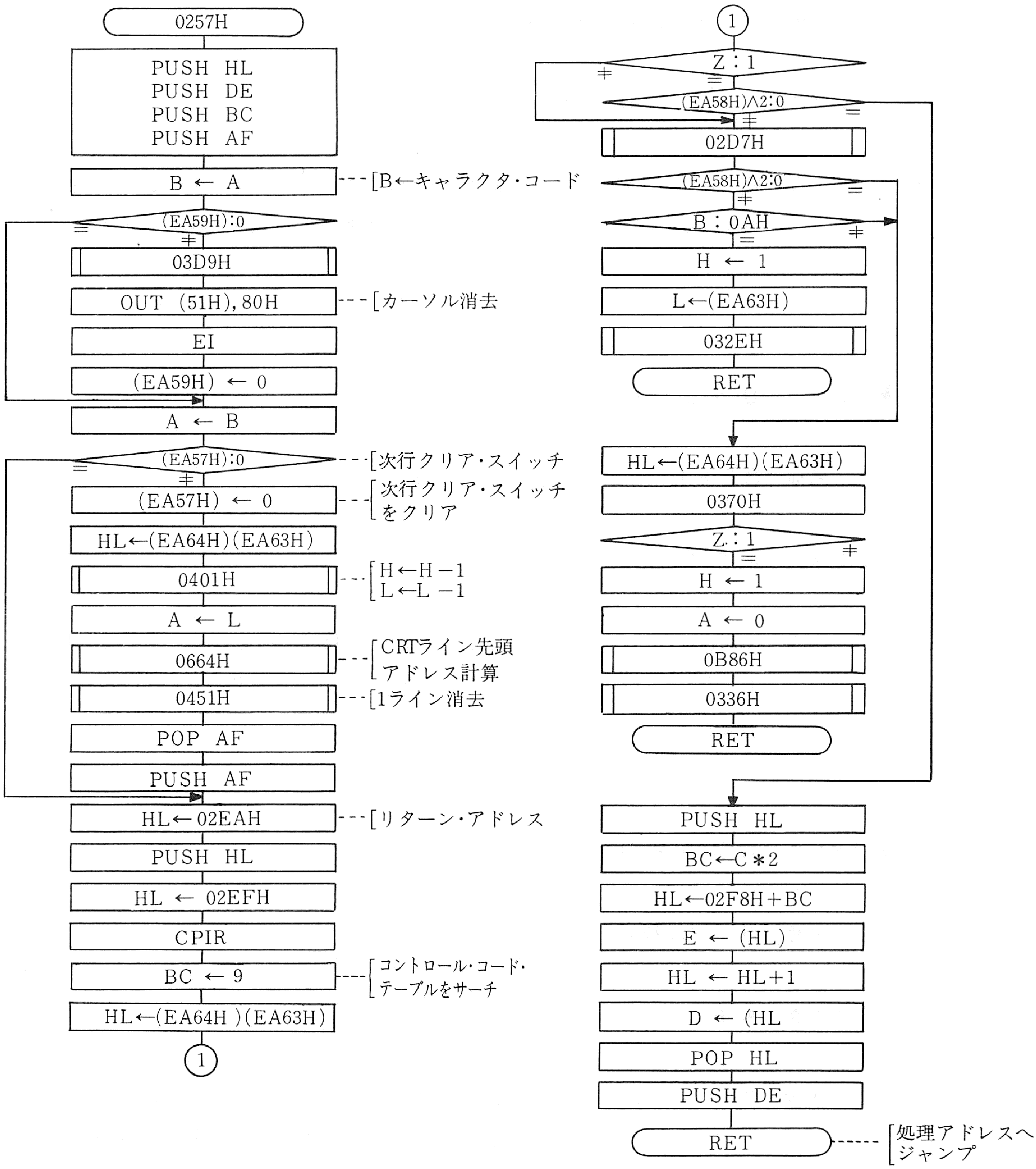
Ａレジスタのキャラクタ１文字を，ＣＲＴ画面上のカーソル位置に表示します.

Ａレジスタのコードが２０Ｈ未満のコントロール・コードであれば，コントロール・コードとしての動作を行ないます.

例　カーソルを画面左上端のホーム・ポジションにもどす.

```
LD      A, 0BH
CALL    0257H
RET
```

フロー・
チャート


0257H
PUSH HL
PUSH DE
PUSH BC
PUSH AF
B ← A
[B←キャラクタ・コード
(EA59H):0
=
≠
03D9H
OUT (51H),80H
[カーソル消去
EI
(EA59H) ← 0
A ← B
(EA57H):0
[次行クリア・スイッチ
=
≠
(EA57H) ← 0
[次行クリア・スイッチをクリア
HL←(EA64H)(EA63H)
0401H
[H←H−1
L←L−1
A ← L
0664H
[CRTライン先頭アドレス計算
0451H
[1ライン消去
POP AF
PUSH AF
HL← 02EAH
[リターン・アドレス
PUSH HL
HL ← 02EFH
CPIR
BC ← 9
[コントロール・コード・テーブルをサーチ
HL←(EA64H)(EA63H)
1
1
Z : 1
≠
=
(EA58H)∧2:0
≠
=
02D7H
(EA58H)∧2:0
≠
=
B : 0AH
=
≠
H ← 1
L←(EA63H)
032EH
RET
HL←(EA64H)(EA63H)
0370H
Z : 1
=
≠
H ← 1
A ← 0
0B86H
0336H
RET
PUSH HL
BC←C＊2
HL←02F8H+BC
E ← (HL)
HL ← HL+1
D ← (HL
POP HL
PUSH DE
RET
[処理アドレスへジャンプ

▶アドレス　０２Ｄ７Ｈ

名称　カーソル移動と１文字表示

レジスタ　A，F，B，C，D，E，H，L

内容　Ｈレジスタの横座標（１～８０）とＬレジスタの縦座標（１～２５）で指定するＣＲＴ画面上の指定位置にカーソルを移動し，Ａレジスタのコード，１バイトを表示します．

▶アドレス　０３５０Ｈ

名称　ＢＥＥＰ

レジスタ　A，F，E，H，L

内容　Ｎ－ＢＡＳＩＣのＢＥＥＰと同じに，一定時間，内蔵スピーカを鳴らします．すでに，スピーカが鳴っている場合には，一定時間，スピーカの音を止めます．

▶アドレス　０３Ａ９Ｈ

名称　カーソル移動

レジスタ　A，F

内容　Ｈレジスタに ＣＲＴ画面上の横座標（１～８０），Ｌレジスタに縦座標（１～２５）を入れてコールすると，カーソルを指定位置に移動する事ができます．

例　N－BASICのLOCATE５，５を行なう.

```
LD      H, 5+1
LD      L, 5+1
CALL    03A9H
RET
```

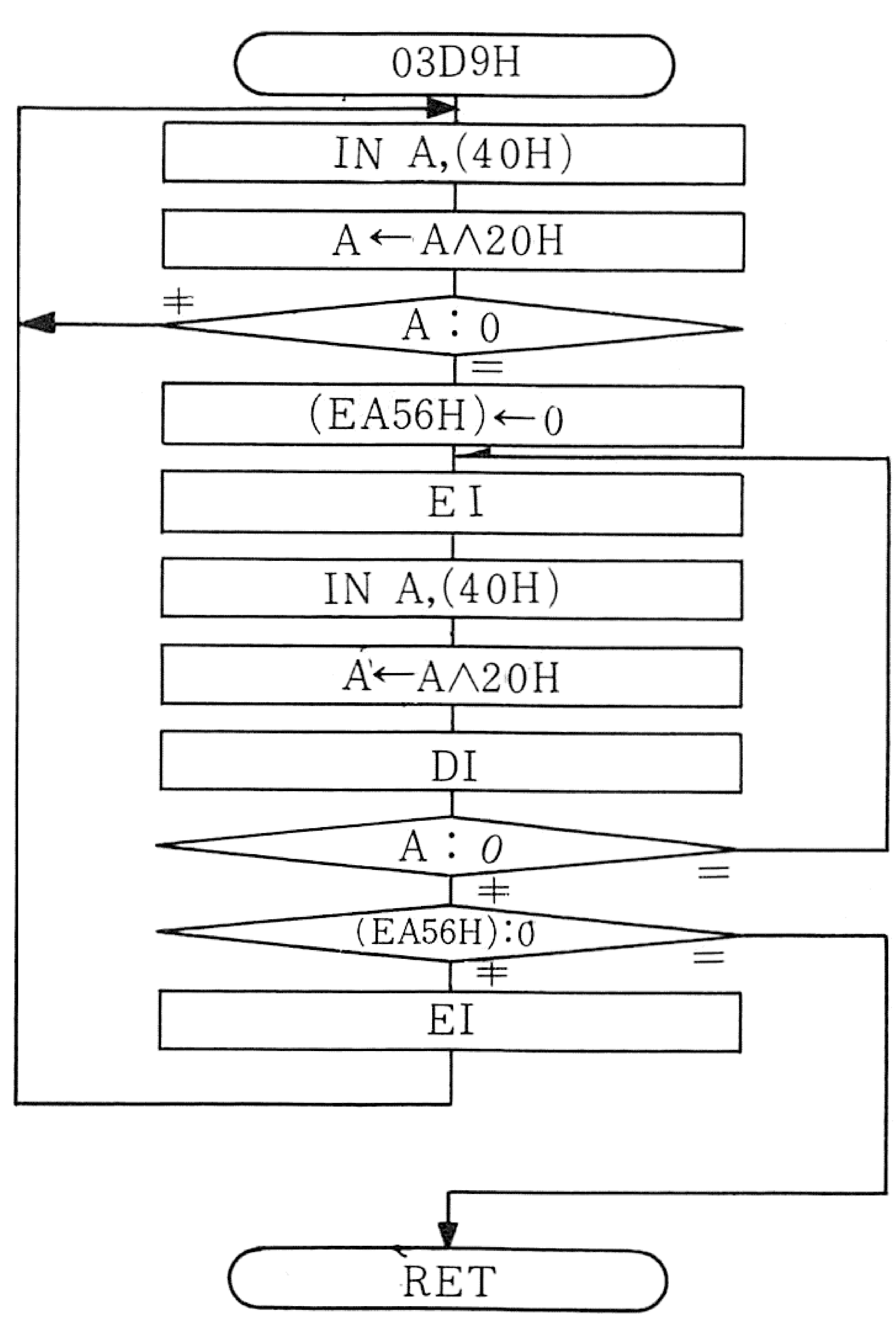
▶アドレス　03D9H

名称　VRTCチェック

レジスタ　A，F

内容　CRTがバーチカル・リトレースになるまでの間，同期待ちを行ないます．VRTCは，入力ポート40H番地の第5ビットによってチェックすることができます．

フロー・チャート



▶アドレス　０３Ｆ３Ｈ

名称　キャラクタ座標→ＶＲＡＭアドレス変換

レジスタ　A，F，D，E，H，L

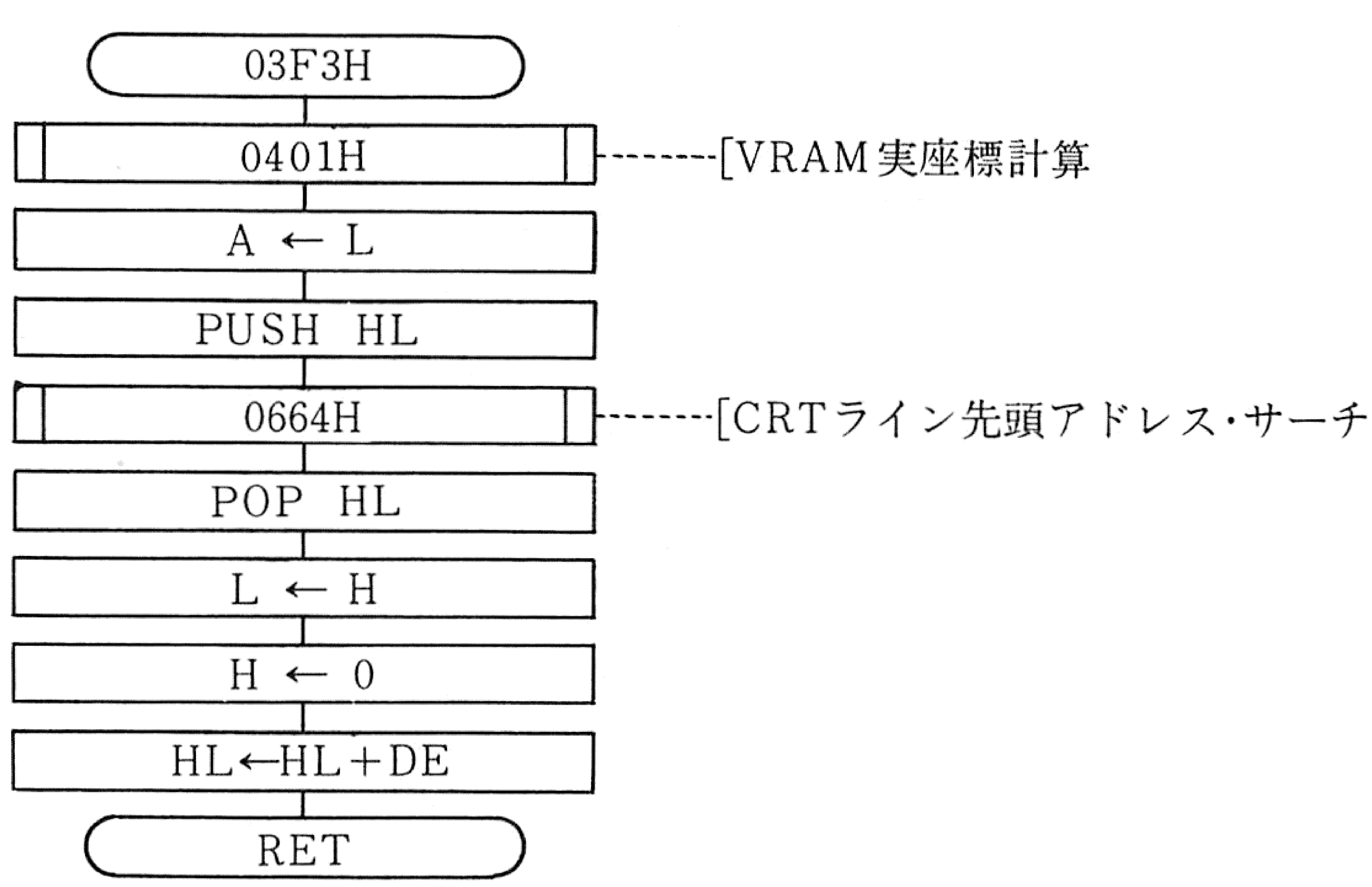
内容　Ｈレジスタに画面上の横座標（１～８０）Ｌレジスタに画面上の縦座標（１～２５）を入れてコールすると，ＶＲＡＭ上のアドレスに変換したものをＨＬレジスタ対に入れてもどります.

８０／４０文字やカラー／白黒の判断も自動的に行なわれます.

例　ホーム・ポジションのアドレスを求める.

```
LD      H, 1
LD      L, 1
CALL    03F3H
RET
```

フロー・チャート



▶アドレス　０４０１H

名称　ＶＲＡＭ実座標計算

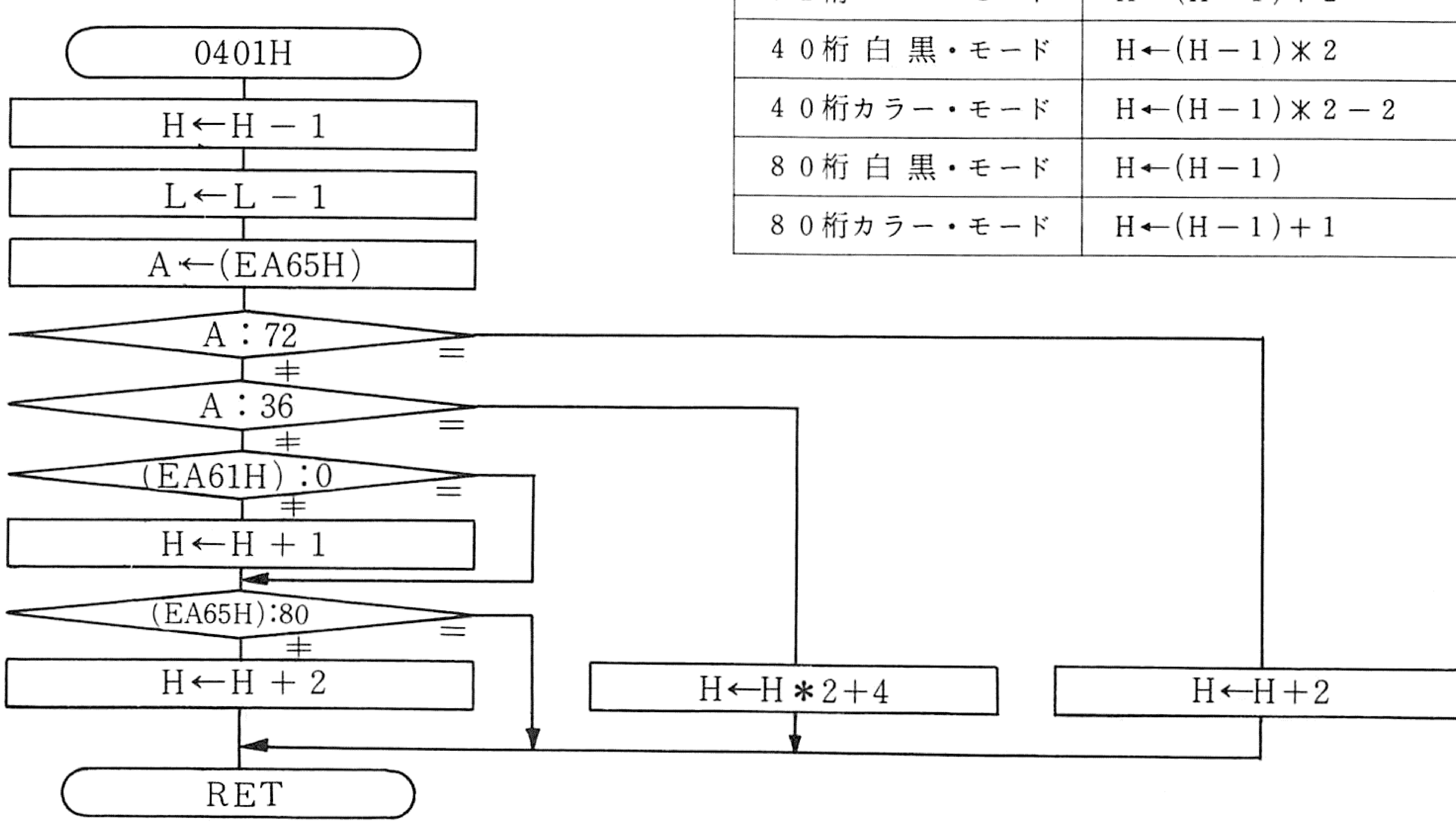
レジスタ　A，F，H，L

内容　Hレジスタの横座標（１～８０）とLレジスタの縦座標（１～２５）を，CRT画面の表示文字数，カラー／白黒モードを考慮した上でVRAM上の実座標に変換し，Hレジスタ（０～７９）とLレジスタ（０～２４）に入れてもどります．

横座標の計算は次のようになります．

モード	計　算
３６桁　モード	H←(H－１)＊２＋４
７２桁　モード	H←(H－１)＋２
４０桁 白 黒・モード	H←(H－１)＊２
４０桁カラー・モード	H←(H－１)＊２－２
８０桁 白 黒・モード	H←(H－１)
８０桁カラー・モード	H←(H－１)＋１

フロー・チャート



▶アドレス　０４２８H

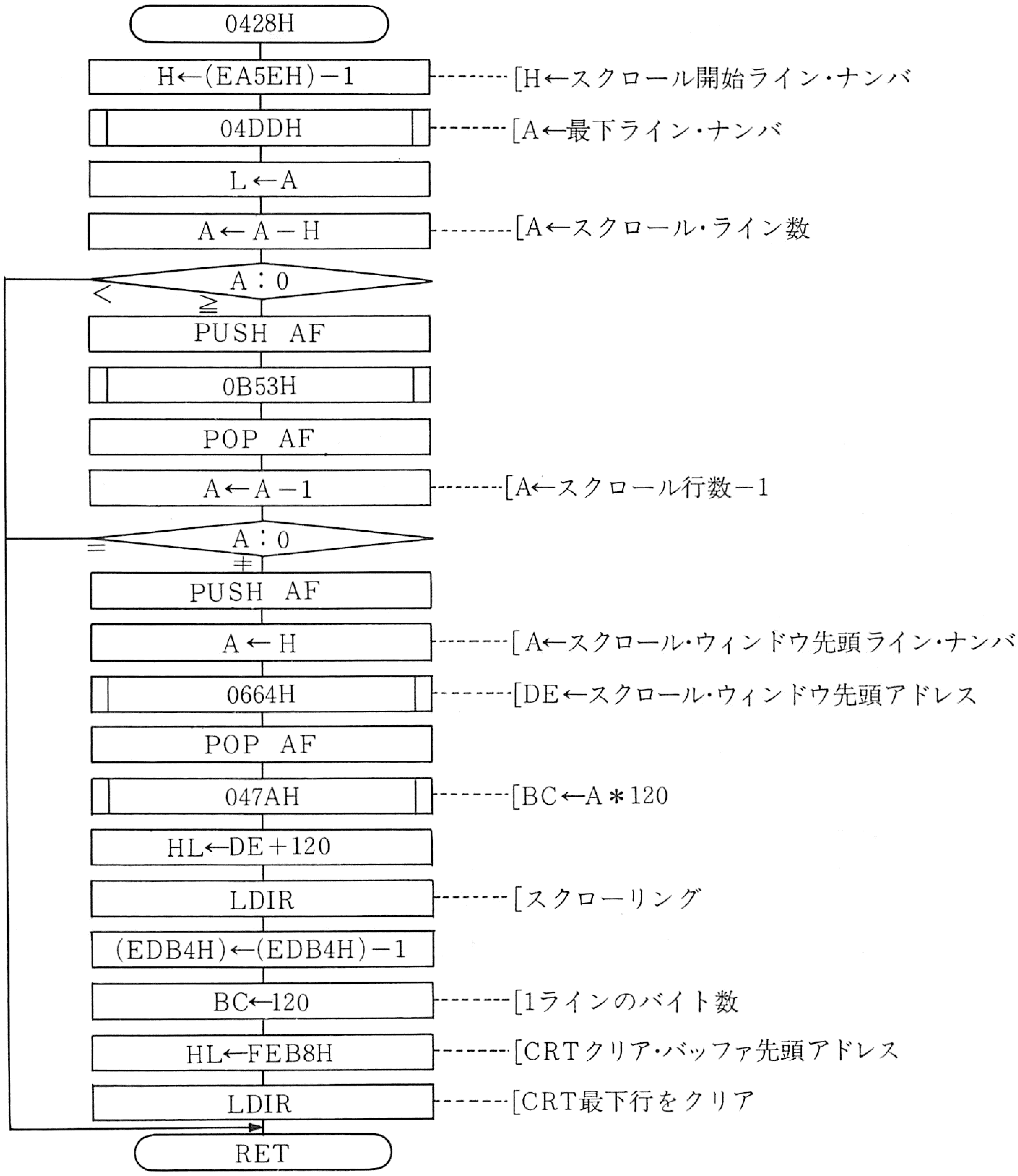
名称　ＣＲＴスクロール

レジスタ　A，F，B，C，D，E，H，L

内容　ワーキング・エリアのデータにより，スクロール・ウインドウの範囲内でスクロールを行ないます.

スクロールを行なった最下段には，クリア・バッファのデータを転送します.

フロー・チャート



▶アドレス　０４５１Ｈ

名称　１行消去

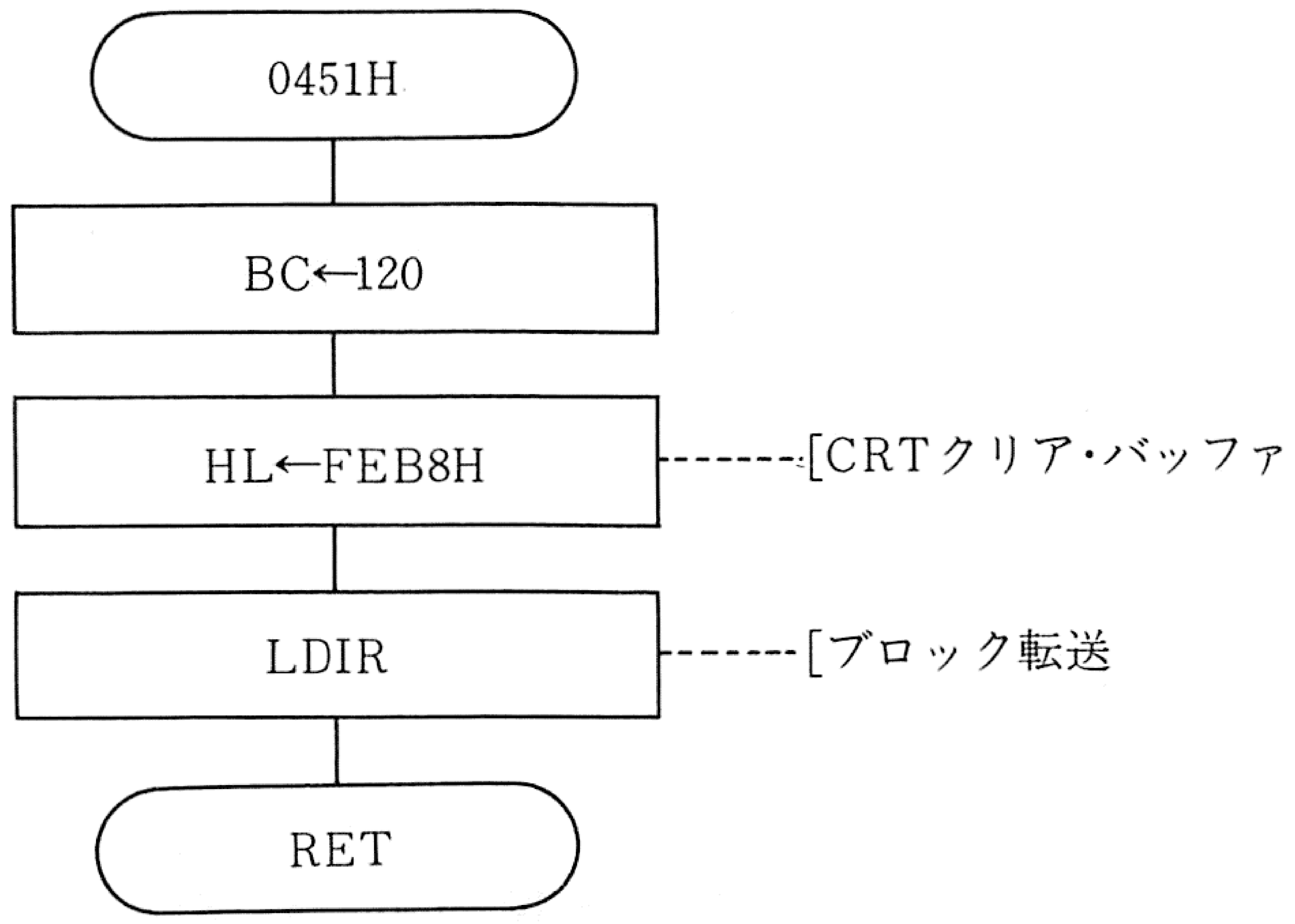
レジスタ　A, F, B, C, D, E, H, L

内容　ＤＥレジスタ対に消去する行の先頭アドレスを入れてコールすると，ＣＲＴ画面上の１行が消去されます．
ヌル・キャラクタ・コード，ベース・カラーなどは，ＦＥＢ８Ｈ番地から１２０バイトのクリア・バッファの値が参照されます．

例　ＣＲＴ画面最上段を消去する．

```
LD      DE, F300H
CALL    0451H
RET
```

フロー・チャート



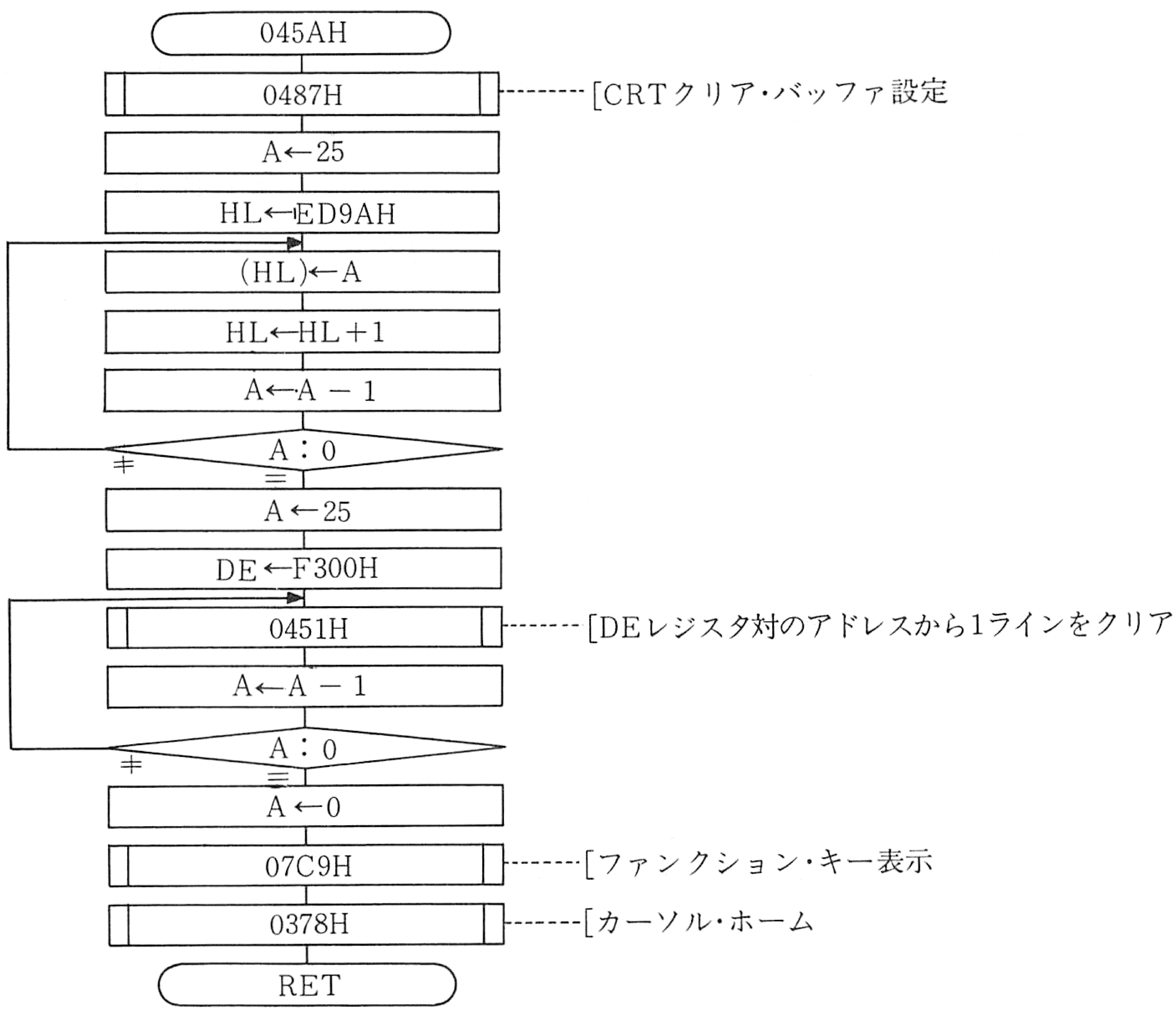
▶アドレス　０４５ＡＨ

名称　画面クリア

レジスタ　A，F，B，C，D，E，H，L

内容　ＣＲＴ画面をクリアします．

フロー・
チャート



▶アドレス　０４７ＡＨ

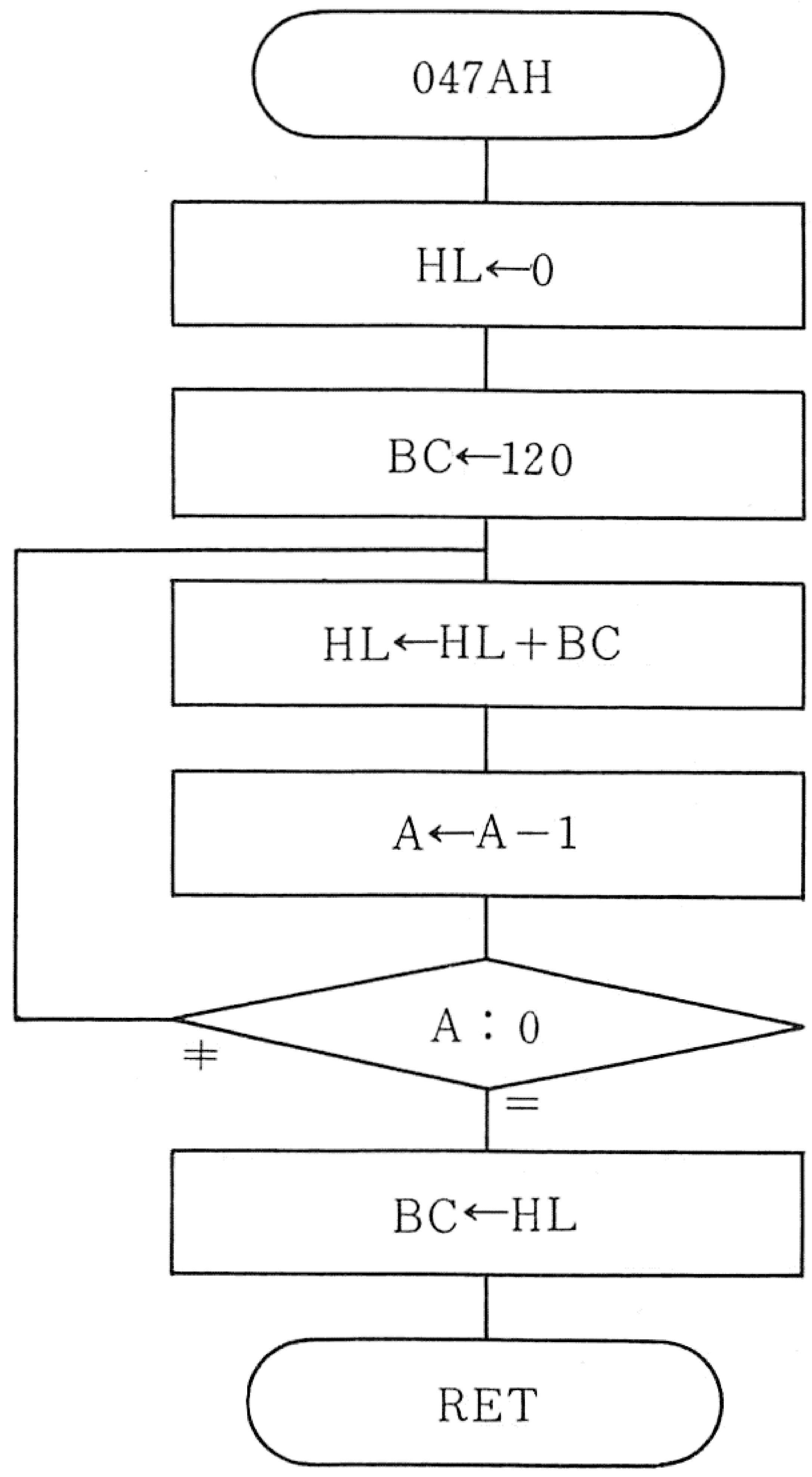
名称　ＣＲＴライン数→ＶＲＡＭバイト数変換

レジスタ　A，F，B，C，H，L

内容　Ａレジスタの値に１２０を乗じたものをＢＣレジスタ対とＨＬレジスタ対に求めます．

Ｎ－ＢＡＳＩＣでは，ＣＲＴ画面のライン数からＶＲＡＭのバイト数を求めるために用いています.

フロー・チャート



▶アドレス　０４８７Ｈ

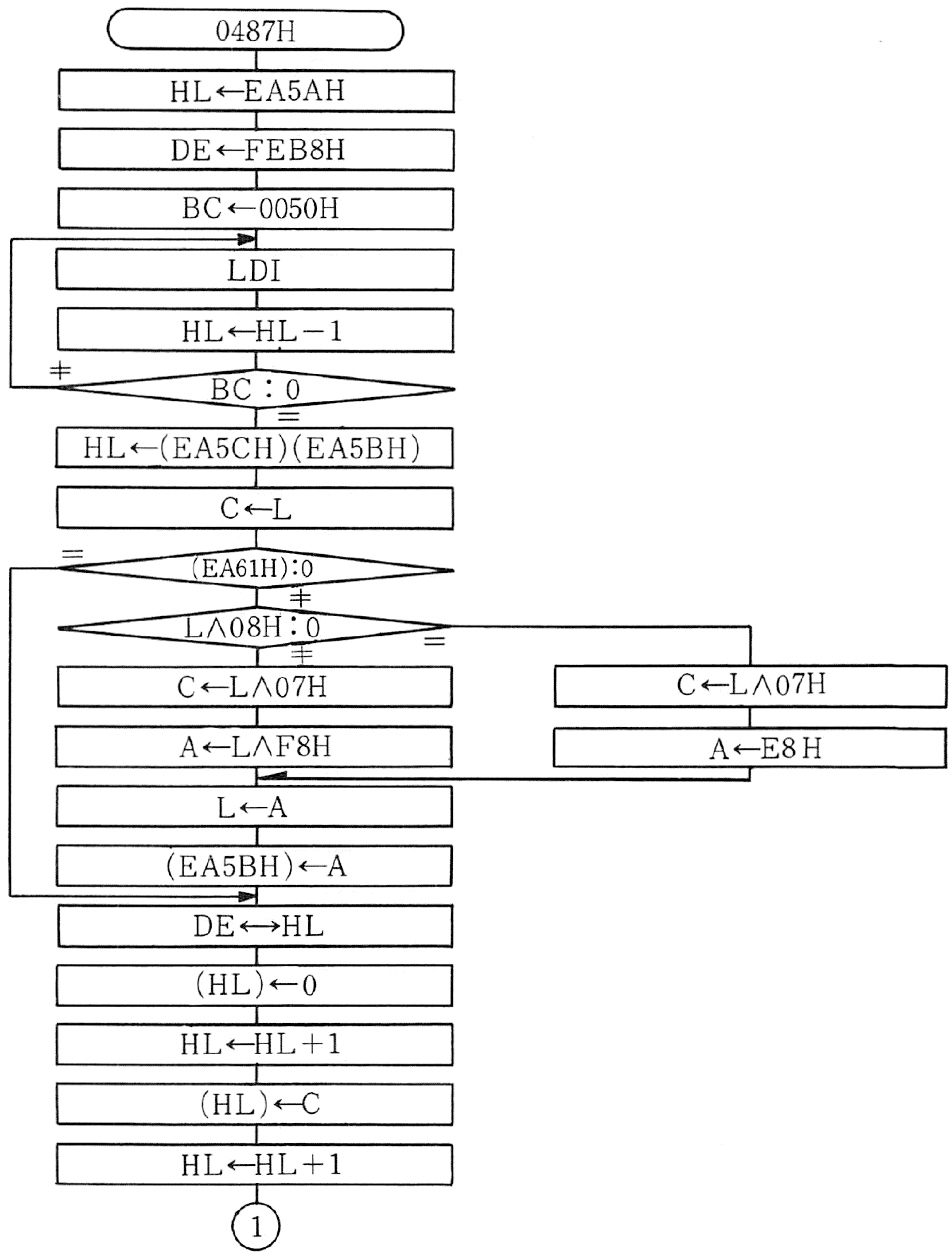
名称　ＣＲＴクリア・バッファの設定

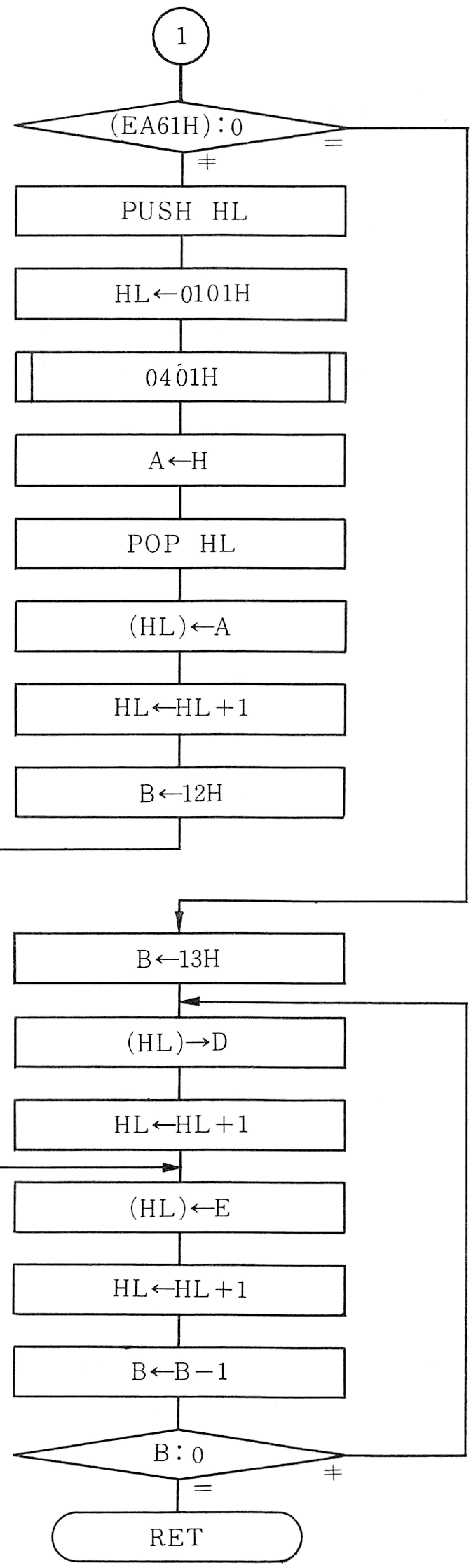
レジスタ　A，F，B，C，D，E，H，L

内容　ＦＥＢ８Ｈ番地から１２０バイトのＣＲＴクリア用のバッファを設定します.

ヌル・キャラクタ・コードおよびベース・カラーはワーキング・エリアのデータを参照して決定します.

フロー・
チャート


0487H
HL←EA5AH
DE←FEB8H
BC←0050H
LDI
HL←HL−1
BC：0
≠
=
HL←(EA5CH)(EA5BH)
C←L
(EA61H)：0
=
≠
L∧08H：0
≠
=
C←L∧07H
A←L∧F8H
C←L∧07H
A←E8H
L←A
(EA5BH)←A
DE←→HL
(HL)←0
HL←HL+1
(HL)←C
HL←HL+1
1


1
(EA61H):0
=
≠
PUSH HL
HL←0101H
0401H
A←H
POP HL
(HL)←A
HL←HL+1
B←12H
B←13H
(HL)→D
HL←HL+1
(HL)←E
HL←HL+1
B←B−1
B:0
≠
=
RET

▶アドレス　04F8H

名称　アトリビュート・コントロール

レジスタ　A, F, B, C, D, E, H, L

内容　Hレジスタに CRT画面上の横座標（1～80），Lレジスタに縦座標（1～25），Aレジスタにアトリビュート・コードを入れてコールすると，アトリビュート・エリアを設定する事ができます.

例　カラー・モード時のホーム・ポジションを緑のキャラクタに設定する.

```
LD    H, 1
LD    L, 1
LD    A, 88H      ; 緑のアトリビュート・コード
CALL  04F8H
RET
```

▶アドレス　0664H

名称　CRTライン先頭アドレス・サーチ

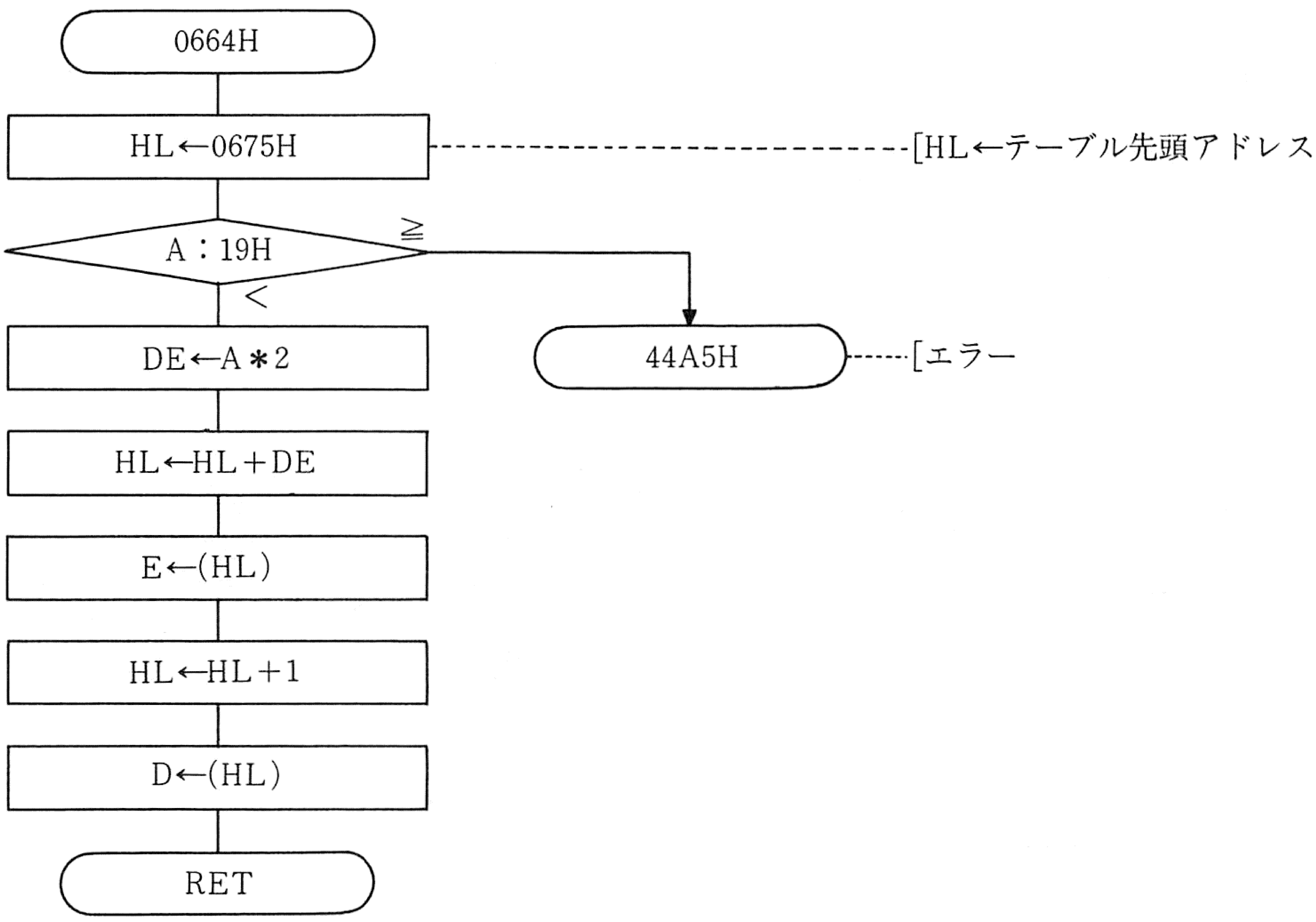
レジスタ　A, F, D, E, H, L

内容　Aレジスタに CRT画面上のライン・ナンバ（0～25）を入れてコールすると，そのラインの先頭アドレスを DEレジスタに求めます.
実際には，演算によってアドレスを求めているのではなく，0675H～06A8H番地のテーブルよりアドレスを引き出しています.

例　CRTの5行目を消去する.

```
LD      A, 5
CALL    0664H
CALL    0451H
RET
```

フロー・
チャート



▶アドレス　０６７５Ｈ～０６Ａ８Ｈ　（ＤＡＴＡ）

名称　ＣＲＴライン先頭アドレス・データ

レジスタ　――――

内容　ビデオ・ラム（ＶＲＡＭ）の各ラインの先頭アドレス，２６組が収納されており，０６６４Ｈ番地からの，ライン先頭アドレス・サーチ・サブルーチンで利用されるデータです．

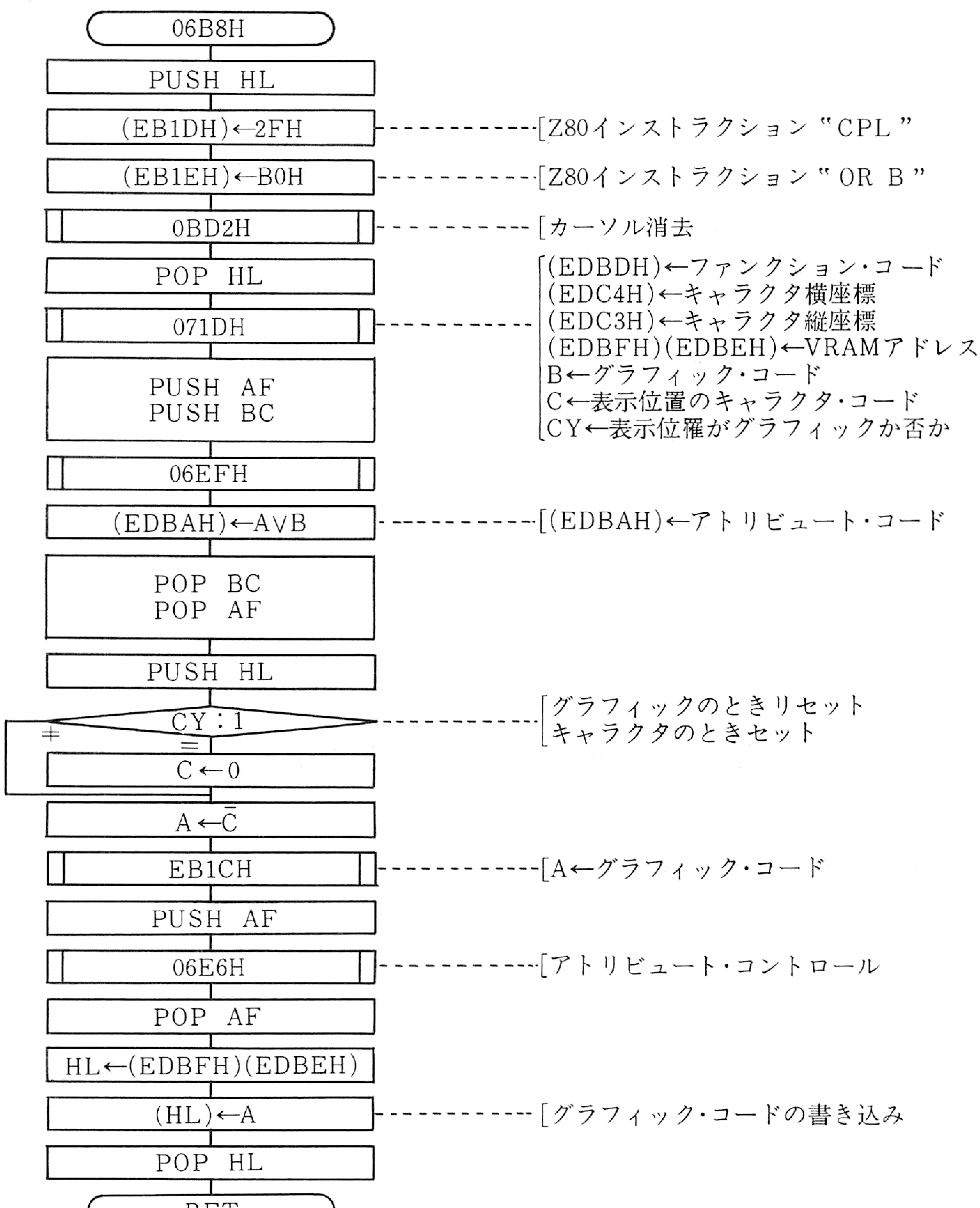
▶アドレス　０６Ｂ８Ｈ

名称　ＰＳＥＴエントリ

レジスタ　――――

内容　N－BASICのPSETエントリ・アドレスです.

フロー・チャート



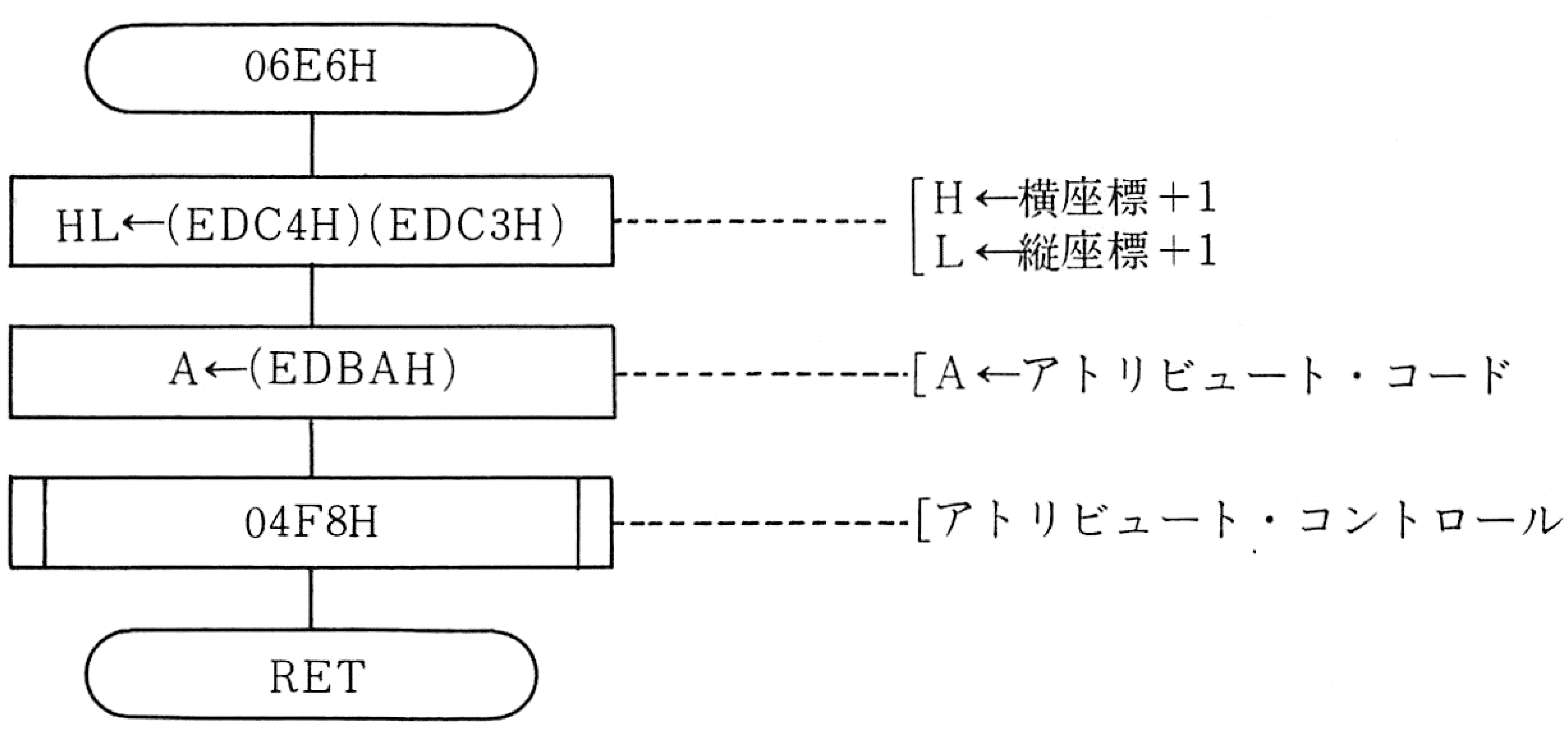
▶アドレス　０６Ｅ６Ｈ

名称　アトリビュート・コントロール

レジスタ　A，F，B，C，D，E，H，L

内容　ＥＤＣ４Ｈ番地にＣＲＴ画面上の横座標（１～８０），ＥＤＣ３Ｈ番地に縦座標（１～２５），ＥＤＢＡＨ番地にアトリビュート・コードを入れてコールすると，アトリビュート・エリアを設定する事ができます．

フロー・チャート



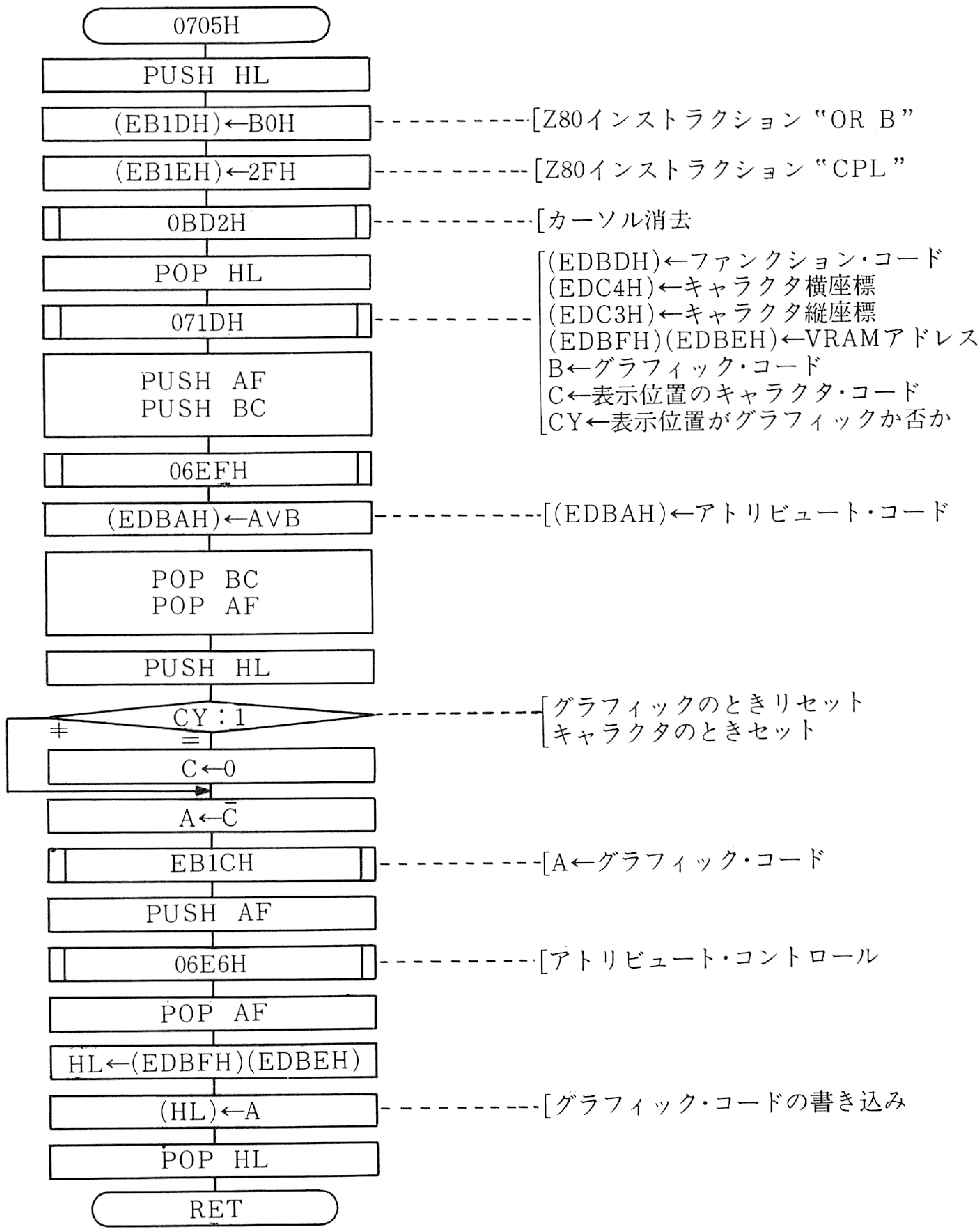
▶アドレス　０７０５Ｈ

名称　ＰＲＥＳＥＴエントリ

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのＰＲＥＳＥＴエントリ・アドレスです．

フロー・
チャート



▶アドレス　071DH

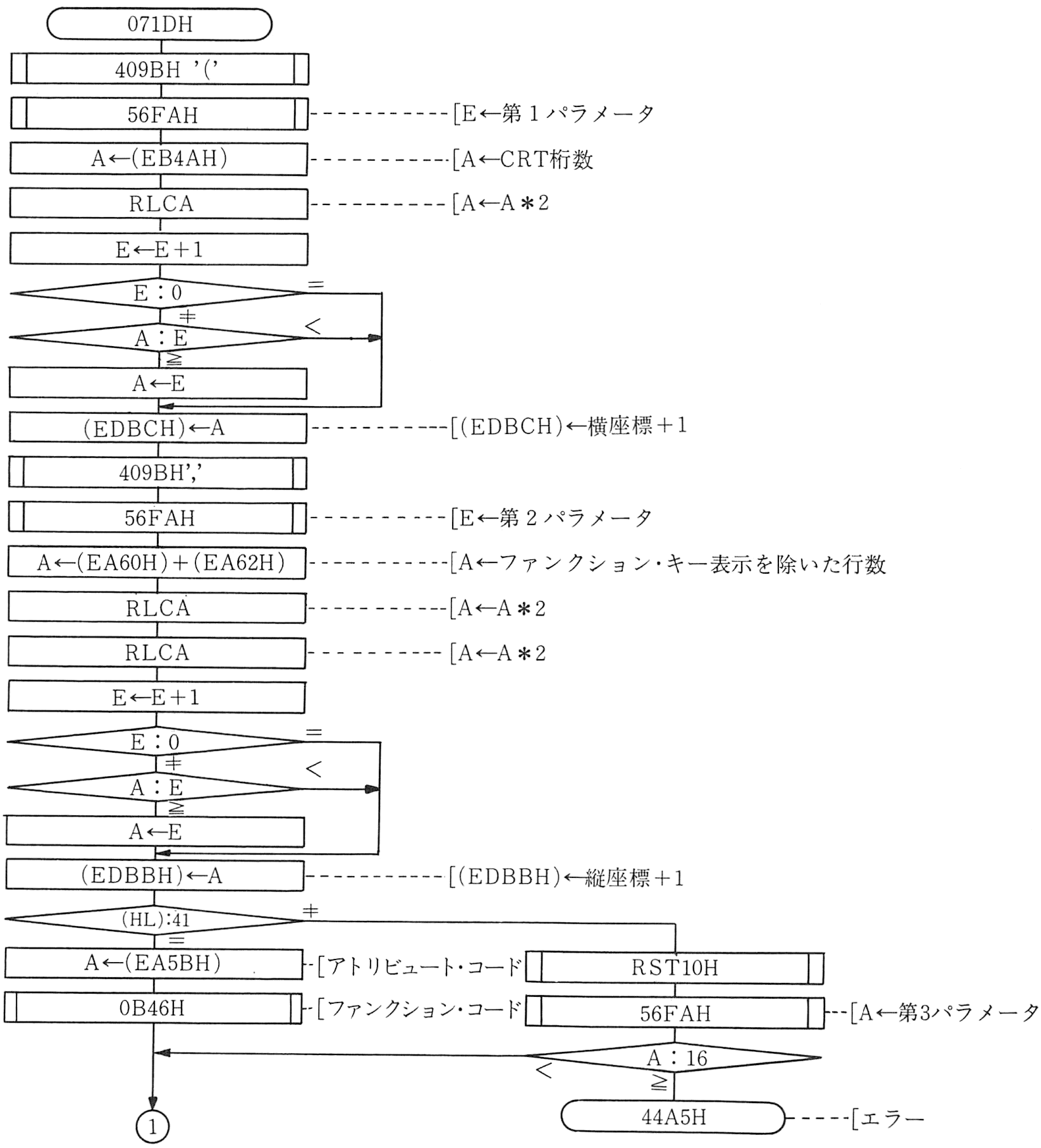
名称　PSET，PRESET用サブルーチン

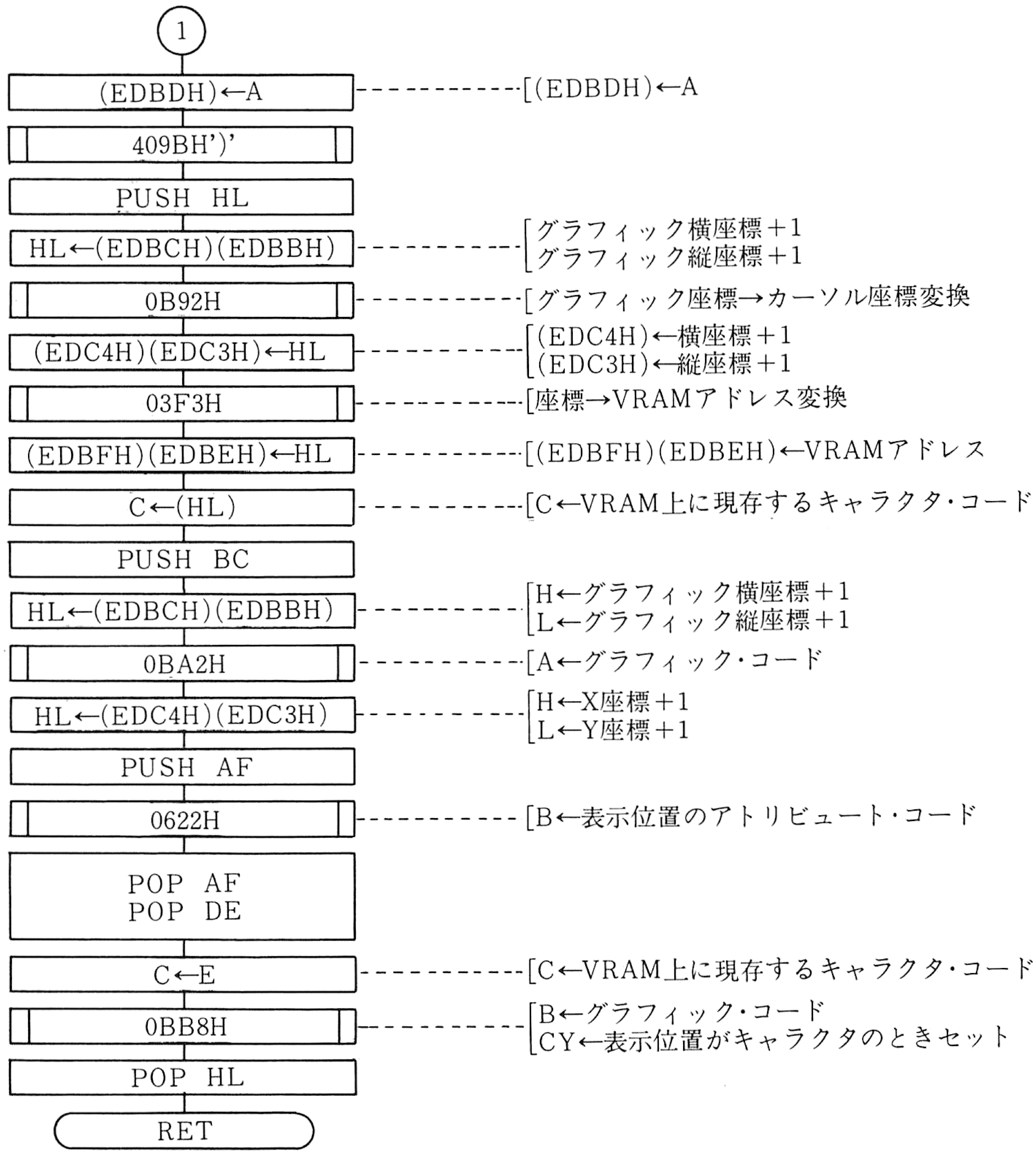
レジスタ　A，F，B，C，D，E，H，L

内容　N－BASICのPSET，PRESETから呼ばれるサブルーチンで，全てのパラメータを受け取った後，レジスタおよびワーキング・エリアを適当な値にセットしてもどります．

詳しくはフロー・チャートを参照してください．

フロー・チャート





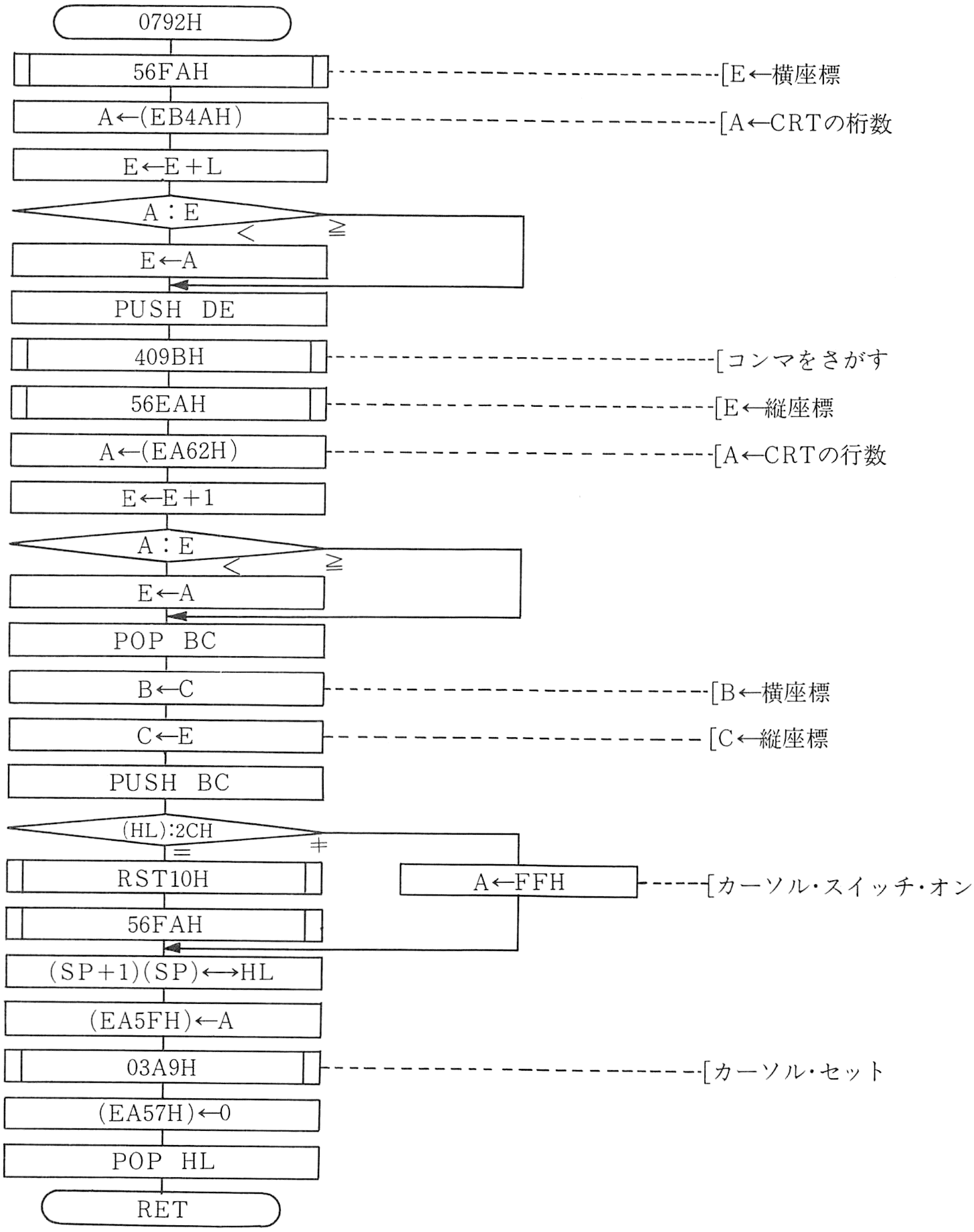
▶アドレス　０７９２Ｈ

名称　ＬＯＣＡＴＥエントリ

レジスタ　————

内容　N－ＢＡＳＩＣのＬＯＣＡＴＥエントリ・アドレスです.

フロー・
チャート



▶アドレス　０７Ｃ９Ｈ

名称　ファンクション・キー表示

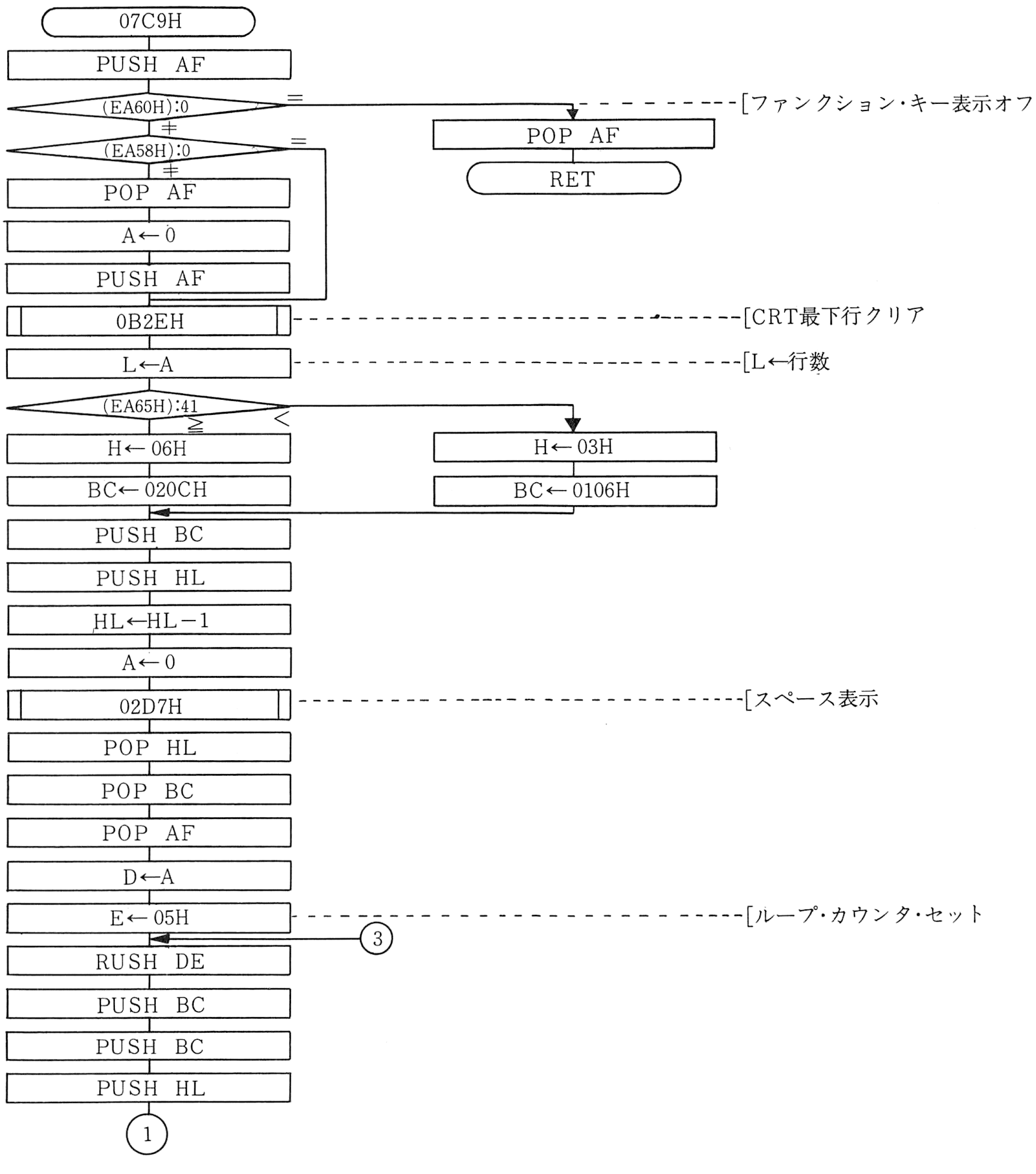
レジスタ　A, F, B, C, D, E, H, L

内容　Ａレジスタのナンバに１を加えたナンバのファンクション・キーから５個のファンクション・キーの内容を，ＣＲＴ画面上最下段の定位置に表示します．ただし，あらかじめファンクション・キー表示モードになっていなければなりません．

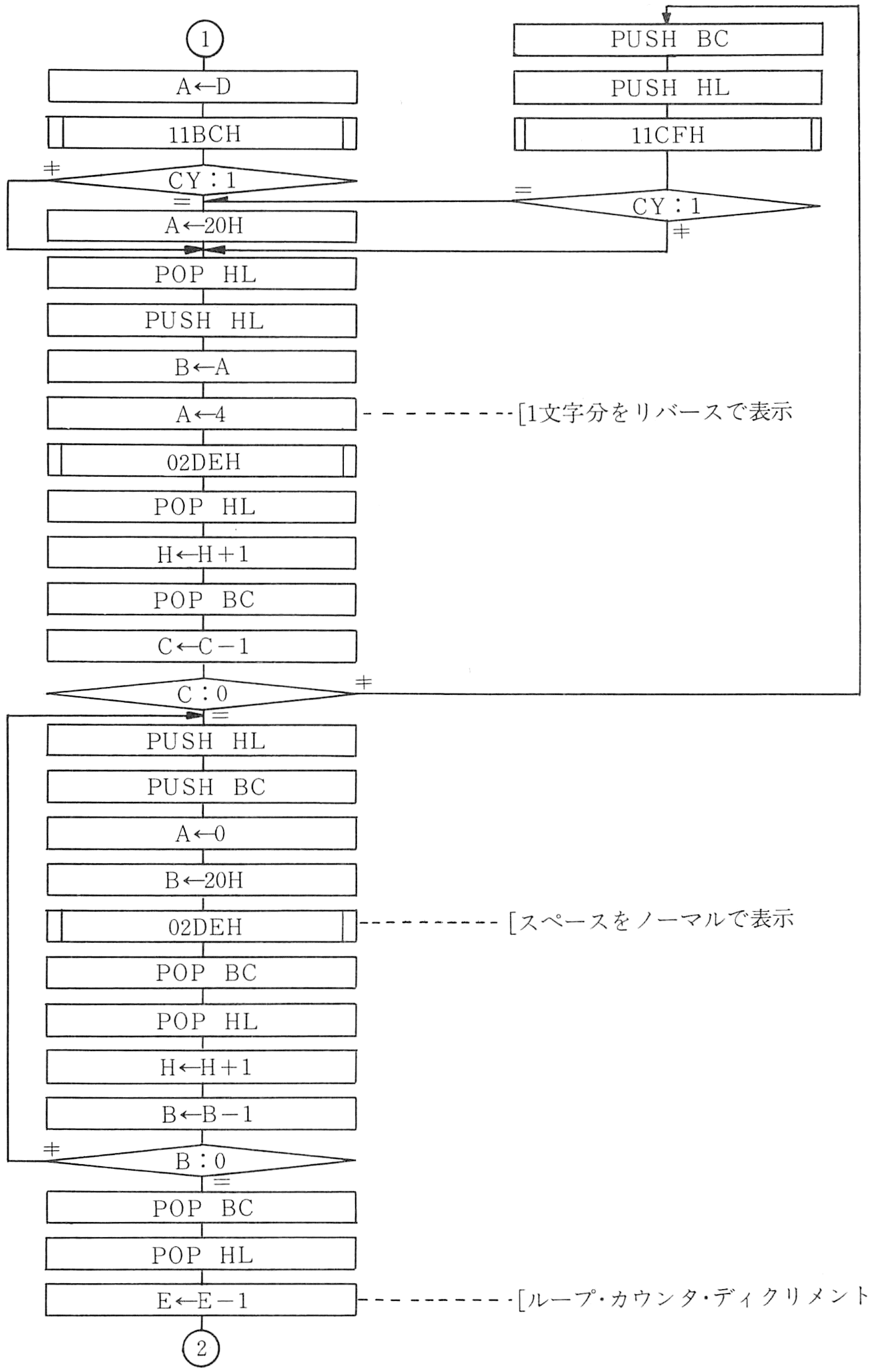
例　[f・3]キーから[f・7]キーまでの内容を表示します．

```
LD     A, 2
CALL   07C9H
RET
```

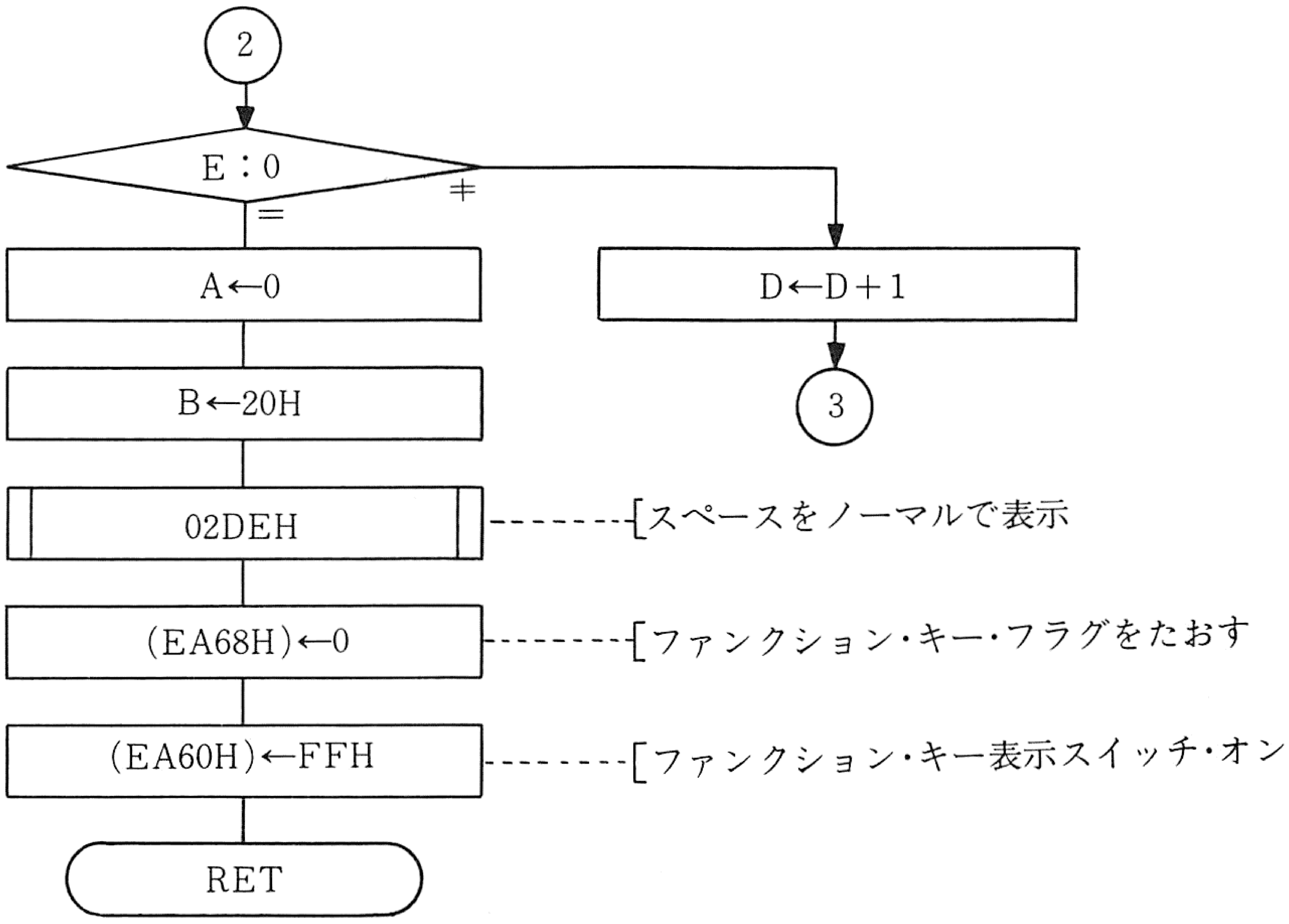
フロー・
チャート



07C9H
PUSH AF
(EA60H):0
=
≠
[ファンクション・キー表示オフ
POP AF
RET
(EA58H):0
=
≠
POP AF
A←0
PUSH AF
0B2EH
[CRT最下行クリア
L←A
[L←行数
(EA65H):41
<
≧
H←06H
BC←020CH
H←03H
BC←0106H
PUSH BC
PUSH HL
HL←HL−1
A←0
02D7H
[スペース表示
POP HL
POP BC
POP AF
D←A
E←05H
[ループ・カウンタ・セット
3
RUSH DE
PUSH BC
PUSH BC
PUSH HL
1



1
A←D
11BCH
CY：1
≠
=
A←20H
POP HL
PUSH HL
B←A
A←4
[1文字分をリバースで表示
02DEH
POP HL
H←H+1
POP BC
C←C−1
C：0
≠
=
PUSH BC
PUSH HL
11CFH
CY：1
=
≠
PUSH HL
PUSH BC
A←0
B←20H
02DEH
[スペースをノーマルで表示
POP BC
POP HL
H←H+1
B←B−1
B：0
≠
=
POP BC
POP HL
E←E−1
[ループ・カウンタ・ディクリメント
2



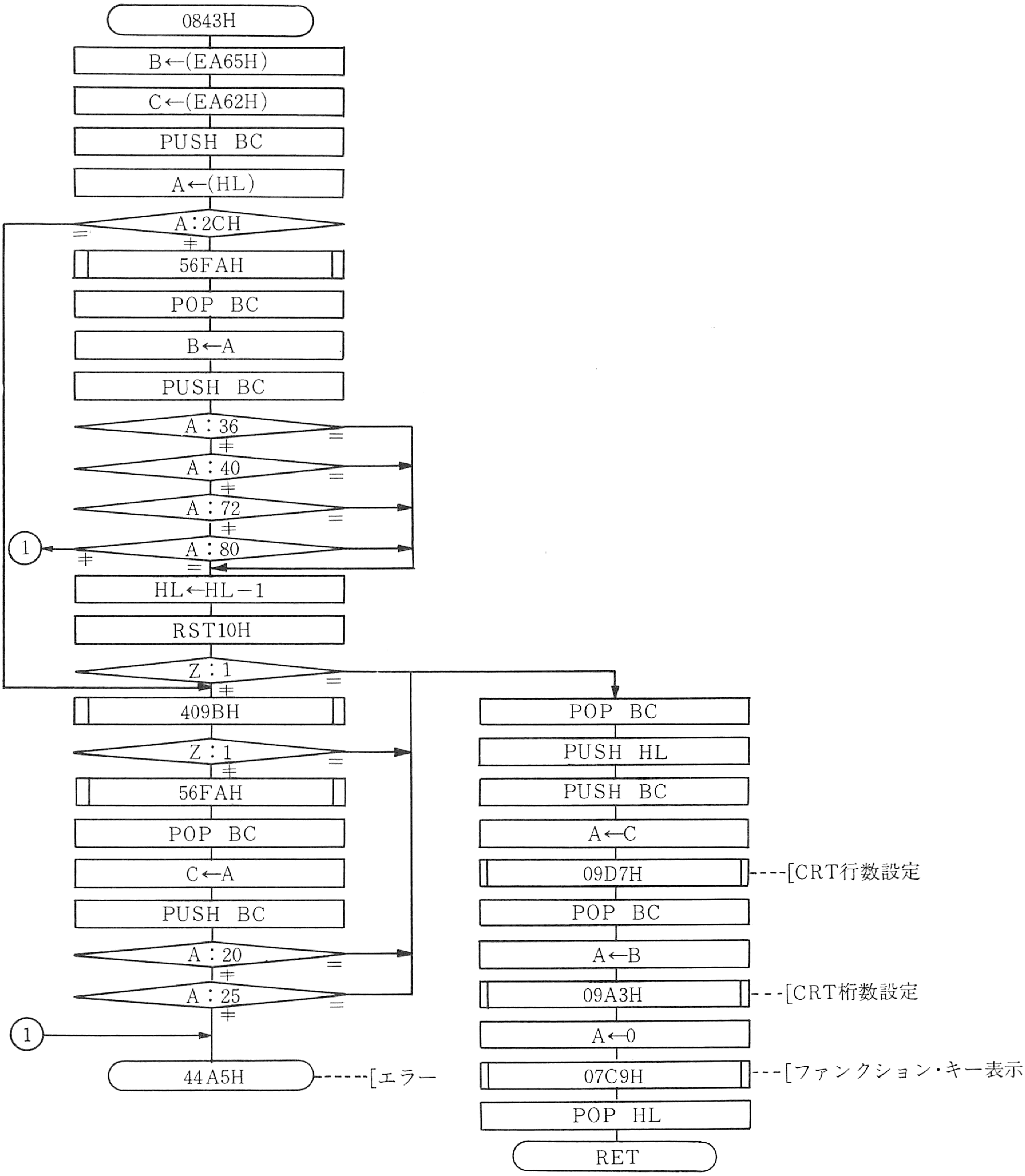
▶アドレス　０８４３Ｈ

名称　ＷＩＤＴＨ

レジスタ　――――

内容　Ｎ―ＢＡＳＩＣのＷＩＤＴＨエントリ・アドレスです．

フロー・
チャート



0843H
B←(EA65H)
C←(EA62H)
PUSH BC
A←(HL)
A:2CH
=
≠
56FAH
POP BC
B←A
PUSH BC
A:36
A:40
A:72
A:80
1
HL←HL−1
RST10H
Z:1
409BH
Z:1
56FAH
POP BC
C←A
PUSH BC
A:20
A:25
44A5H
[エラー
POP BC
PUSH HL
PUSH BC
A←C
09D7H
[CRT行数設定
POP BC
A←B
09A3H
[CRT桁数設定
A←0
07C9H
[ファンクション・キー表示
POP HL
RET

▶アドレス　０８Ｆ７Ｈ

名称　ファンクション・キー表示有無とカラー／白黒切換え

レジスタ　A, F, B, C, D, E

内容　Ｂレジスタにファンクション・キー表示の有／無を，Ｃレジスタにカラー／白黒モード切換えを入れてコールします．
ただし，各パラメータは，Ｎ―ＢＡＳＩＣのような１か０ではなく，ＦＦＨか０を使用しなければなりません．

例　ファンクション・キーの表示を消しカラー・モードに設定する．

```
LD      B, 0
LD      C, FFH
CALL    08F7H
RET
```

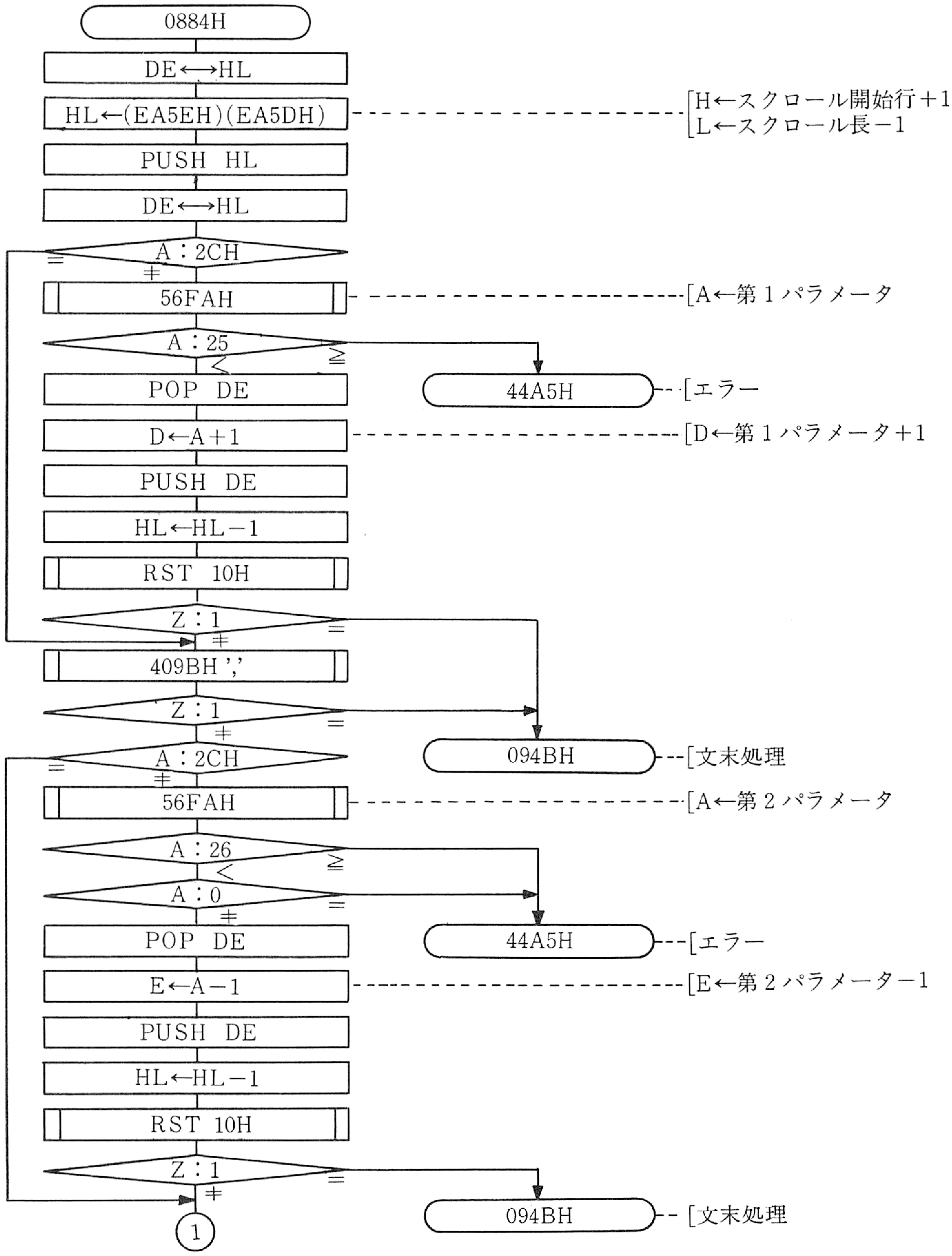
▶アドレス　０８８４Ｈ

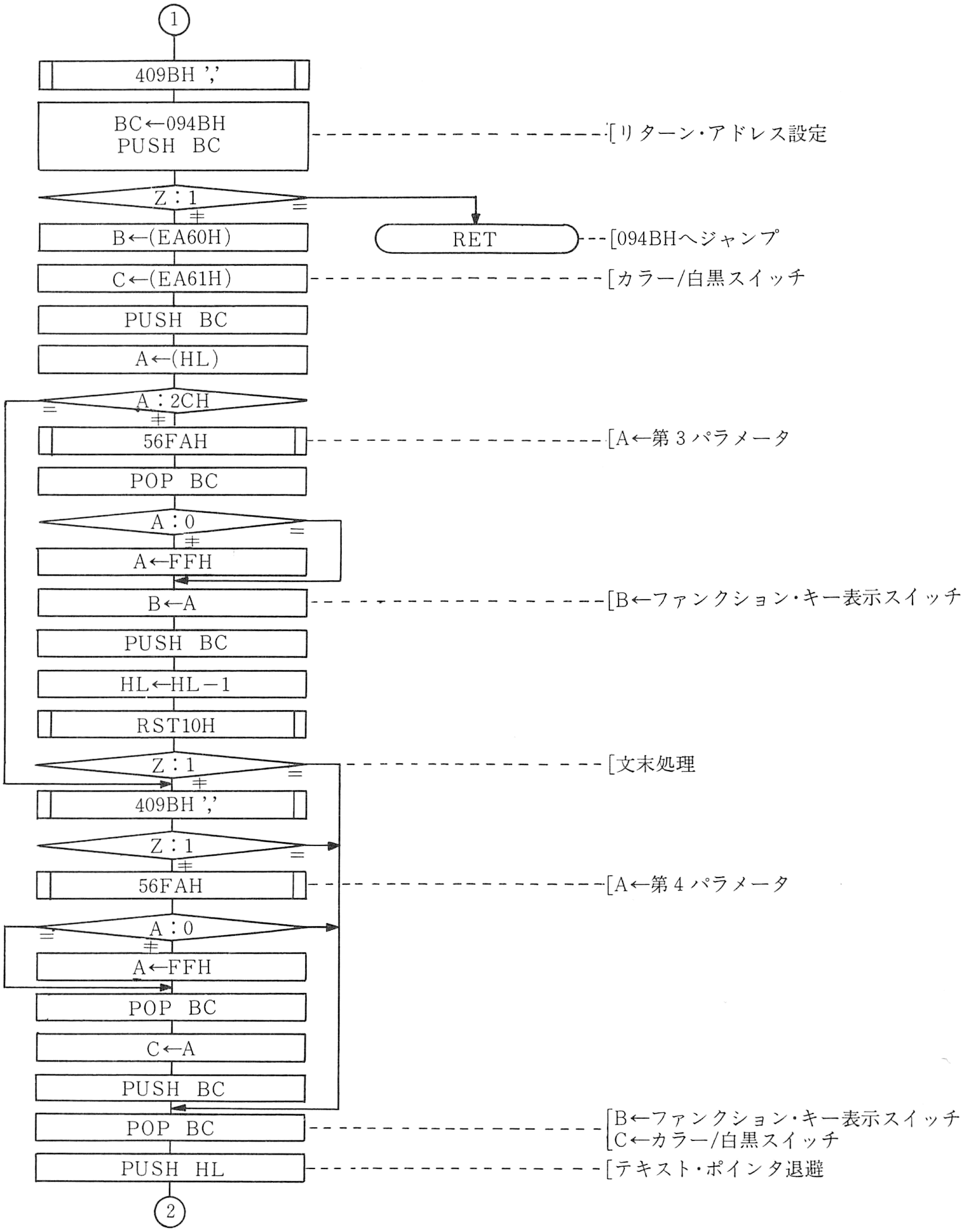
名称　ＣＯＮＳＯＬＥエントリ

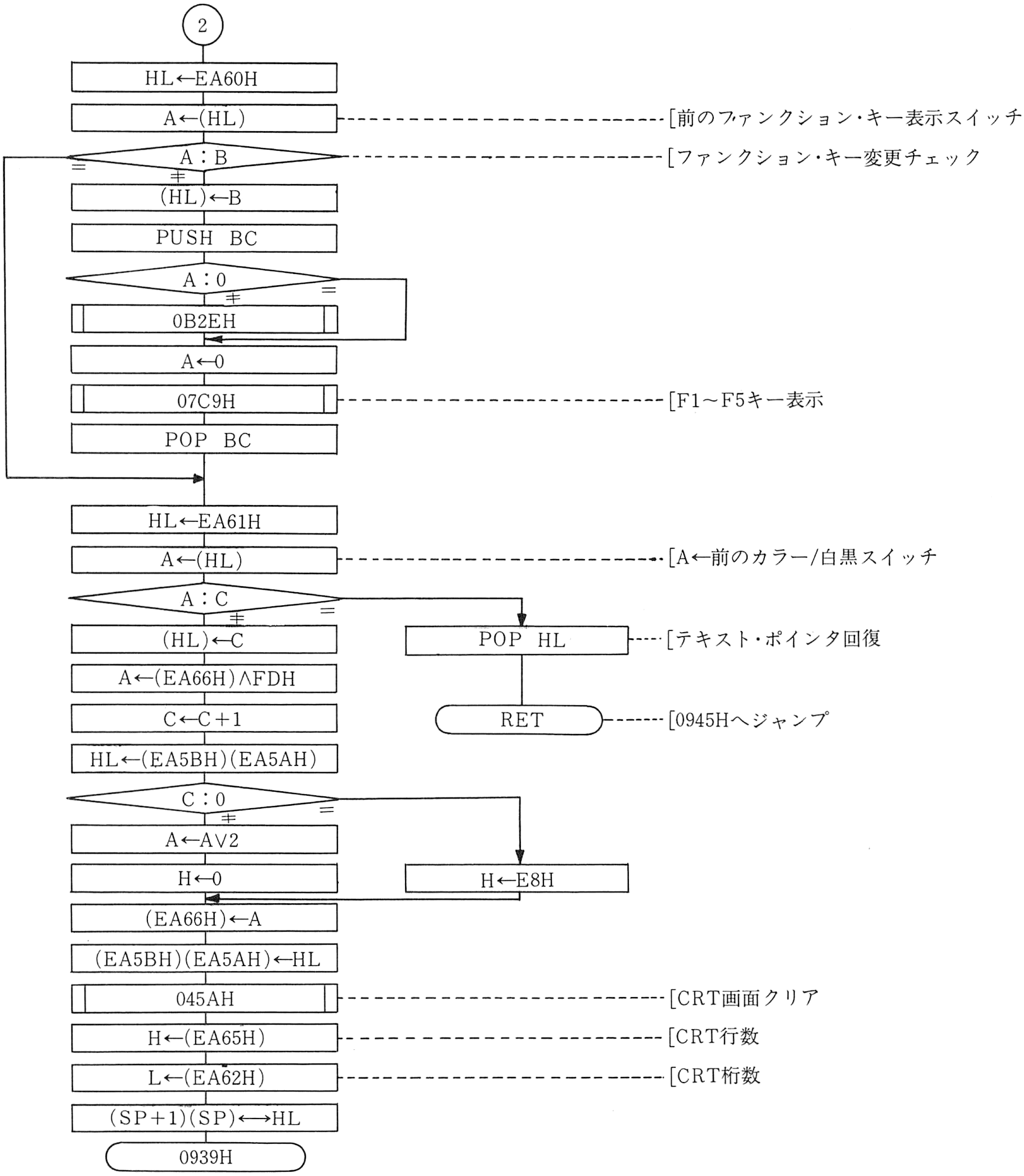
レジスタ　――――

内容　Ｎ―ＢＡＳＩＣのＣＯＮＳＯＬＥエントリアドレスです．

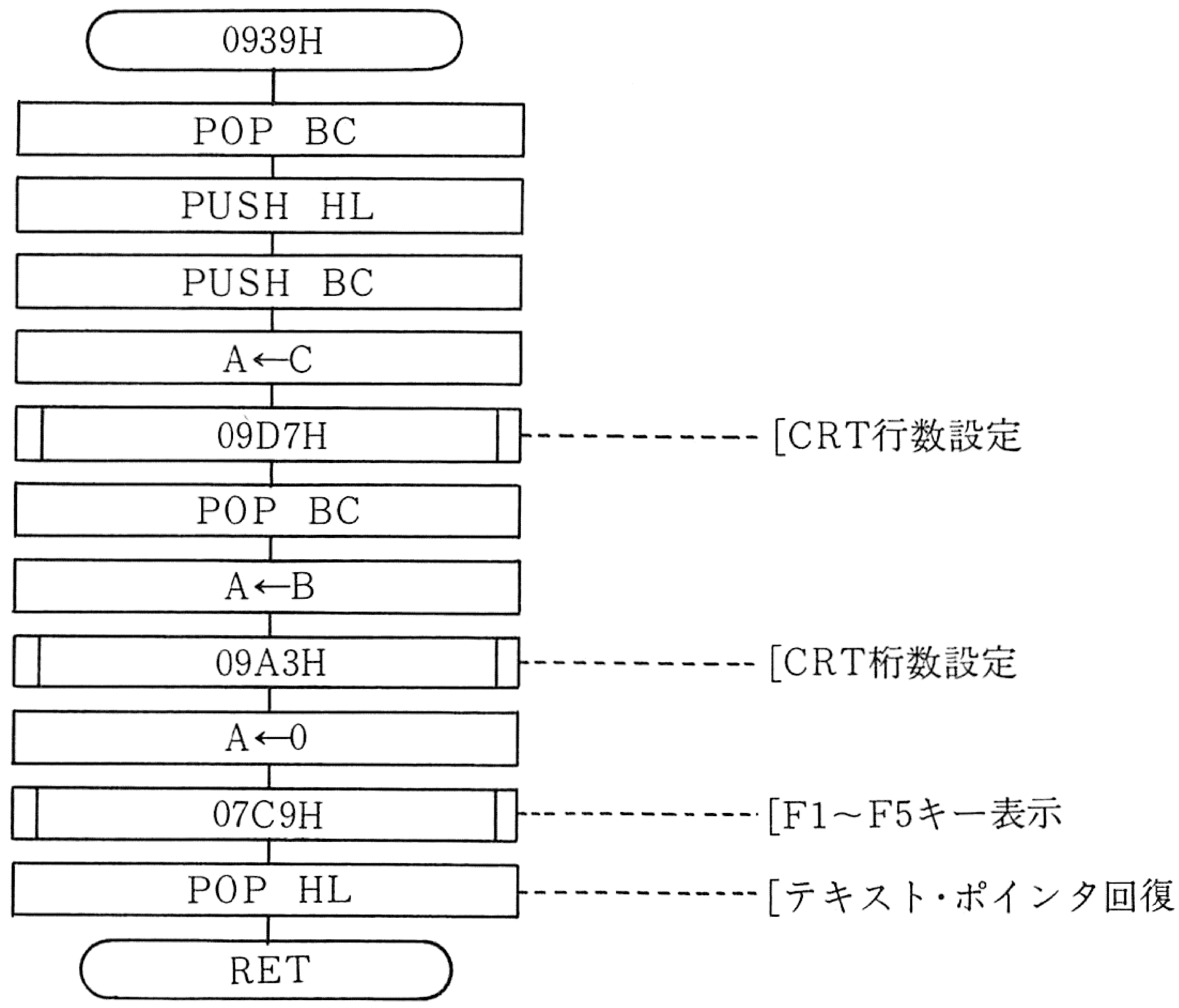
フロー・
チャート


0884H
DE←→HL
HL←(EA5EH)(EA5DH)
H←スクロール開始行+1
L←スクロール長−1
PUSH HL
DE←→HL
A：2CH
=
≠
56FAH
A←第1パラメータ
A：25
<
≧
44A5H
エラー
POP DE
D←A+1
D←第1パラメータ+1
PUSH DE
HL←HL−1
RST 10H
Z：1
=
≠
409BH ','
Z：1
094BH
文末処理
A：2CH
56FAH
A←第2パラメータ
A：26
A：0
44A5H
エラー
POP DE
E←A−1
E←第2パラメータ−1
PUSH DE
HL←HL−1
RST 10H
Z：1
094BH
文末処理
1


1
409BH ','
BC←094BH
PUSH BC
[リターン・アドレス設定
Z：1
=
≠
RET
[094BHへジャンプ
B←(EA60H)
C←(EA61H)
[カラー/白黒スイッチ
PUSH BC
A←(HL)
A：2CH
56FAH
[A←第3パラメータ
POP BC
A：0
A←FFH
B←A
[B←ファンクション・キー表示スイッチ
PUSH BC
HL←HL−1
RST10H
Z：1
[文末処理
409BH ','
Z：1
56FAH
[A←第4パラメータ
A：0
A←FFH
POP BC
C←A
PUSH BC
POP BC
[B←ファンクション・キー表示スイッチ
C←カラー/白黒スイッチ
PUSH HL
[テキスト・ポインタ退避
2



2
HL←EA60H
A←(HL)
[前のファンクション・キー表示スイッチ
A：B
[ファンクション・キー変更チェック
=
≠
(HL)←B
PUSH BC
A：0
≠
=
0B2EH
A←0
07C9H
[F1～F5キー表示
POP BC
HL←EA61H
A←(HL)
[A←前のカラー/白黒スイッチ
A：C
=
≠
(HL)←C
POP HL
[テキスト・ポインタ回復
A←(EA66H)∧FDH
C←C+1
RET
[0945Hへジャンプ
HL←(EA5BH)(EA5AH)
C：0
≠
=
A←A∨2
H←0
H←E8H
(EA66H)←A
(EA5BH)(EA5AH)←HL
045AH
[CRT画面クリア
H←(EA65H)
[CRT行数
L←(EA62H)
[CRT桁数
(SP+1)(SP)←→HL
0939H



▶アドレス　０９３０H

名称　ＣＲＴ画面表示文字数の設定

レジスタ　A，F，B，C，D，E，H，L

内容　ＥＡ６２Ｈ番地に行数，ＥＡ６５Ｈ番地に桁数を入れて，コールする事によって，ＣＲＴ画面に表示する文字数を設定する事ができます．
ＣＲＴコントローラ，及び，ＤＭＡコントローラのイニシアライズ等を行なう場合には単にコールだけしてください．

▶アドレス　０９３AH

名称　ＣＲＴ画面表示文字数の設定

レジスタ　A，F，B，C，D，E

内容　Ｂレジスタに桁数，Ｃレジスタに行数を入れてコールします．
Ｎ－ＢＡＳＩＣのＷＩＤＴＨで不可能な，例えば，横１０桁などに設定する事も可能です．

例　８０桁×２５行に設定する．

```
LD    B, 80
LD    C, 25
CALL  093AH
RET
```

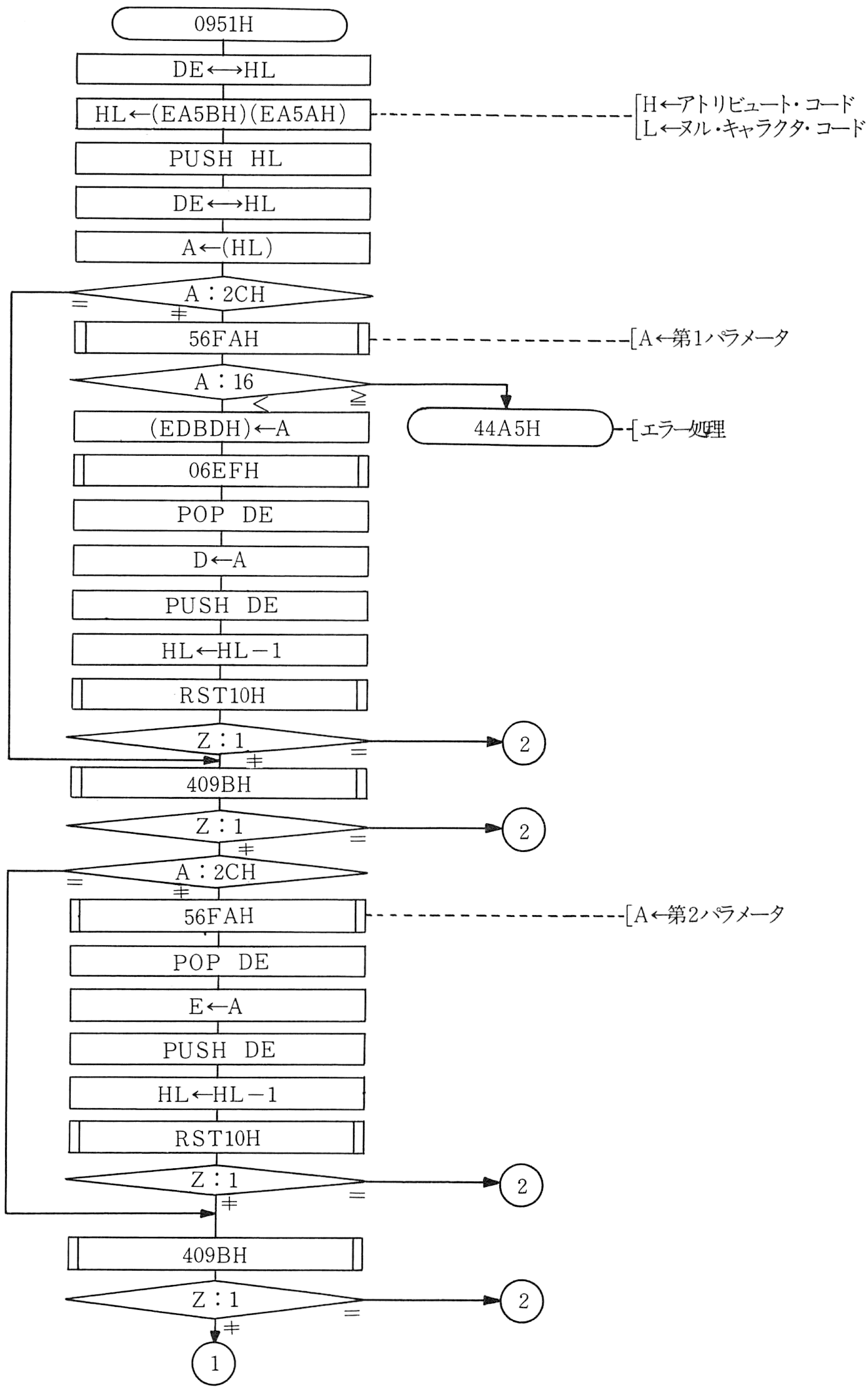
▶アドレス　**０９５１Ｈ**

名称　**ＣＯＬＯＲエントリ**

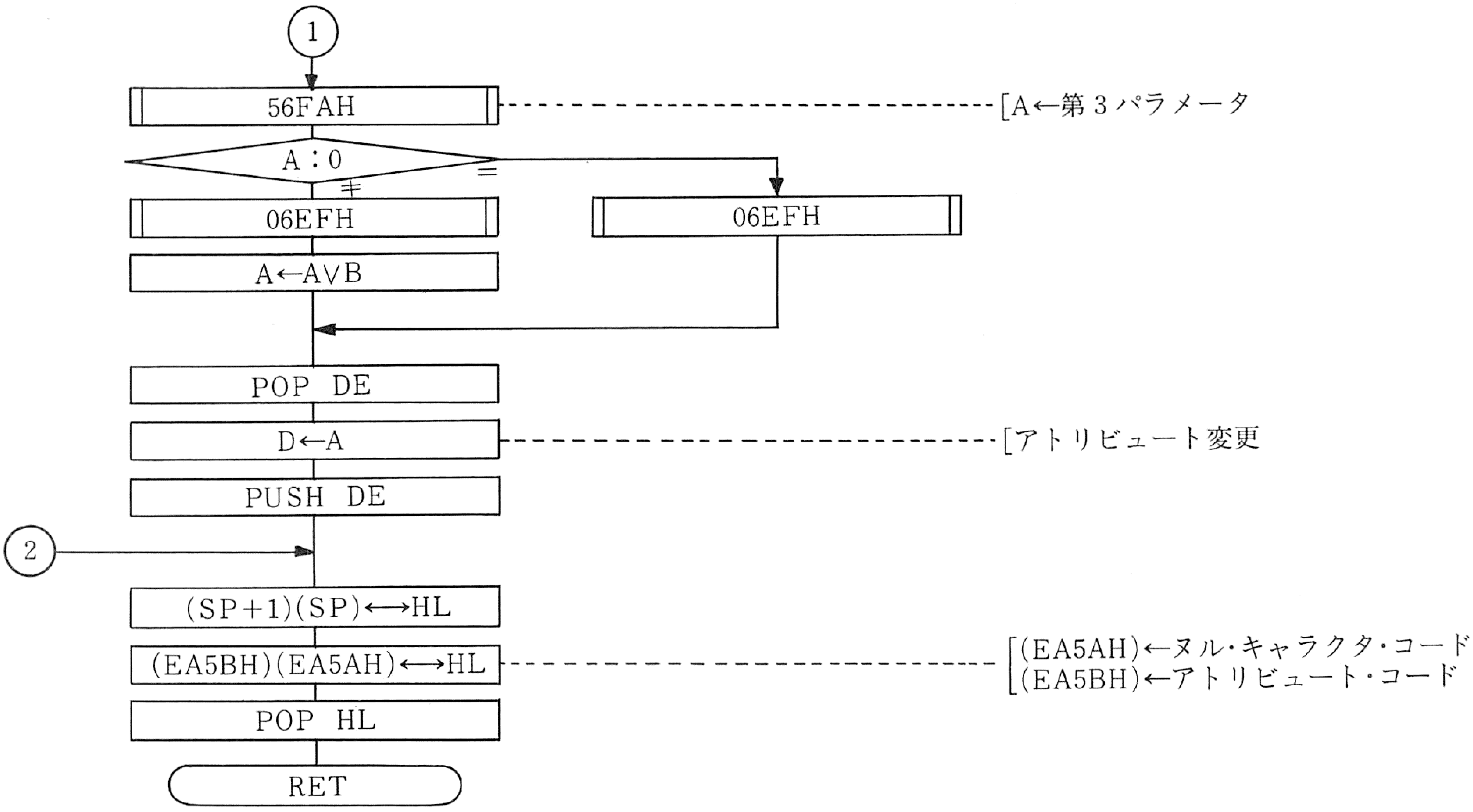
レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのＣＯＬＯＲエントリ・アドレスです．

フロー・
チャート



0951H
DE⟷HL
HL←(EA5BH)(EA5AH)
H←アトリビュート・コード
L←ヌル・キャラクタ・コード
PUSH HL
DE⟷HL
A←(HL)
A：2CH
＝
≠
56FAH
A←第1パラメータ
A：16
<
≧
(EDBDH)←A
44A5H
エラー処理
06EFH
POP DE
D←A
PUSH DE
HL←HL－1
RST10H
Z：1
2
409BH
A：2CH
56FAH
A←第2パラメータ
POP DE
E←A
PUSH DE
HL←HL－1
RST10H
Z：1
409BH
Z：1
1



▶アドレス　０９Ａ３Ｈ

名称　ＣＲＴ画面表示桁数の設定

レジスタ　A, F, B, C, D, E, H, L

内容　Ａレジスタに桁数を入れてコールします.

例　８０桁に設定する.

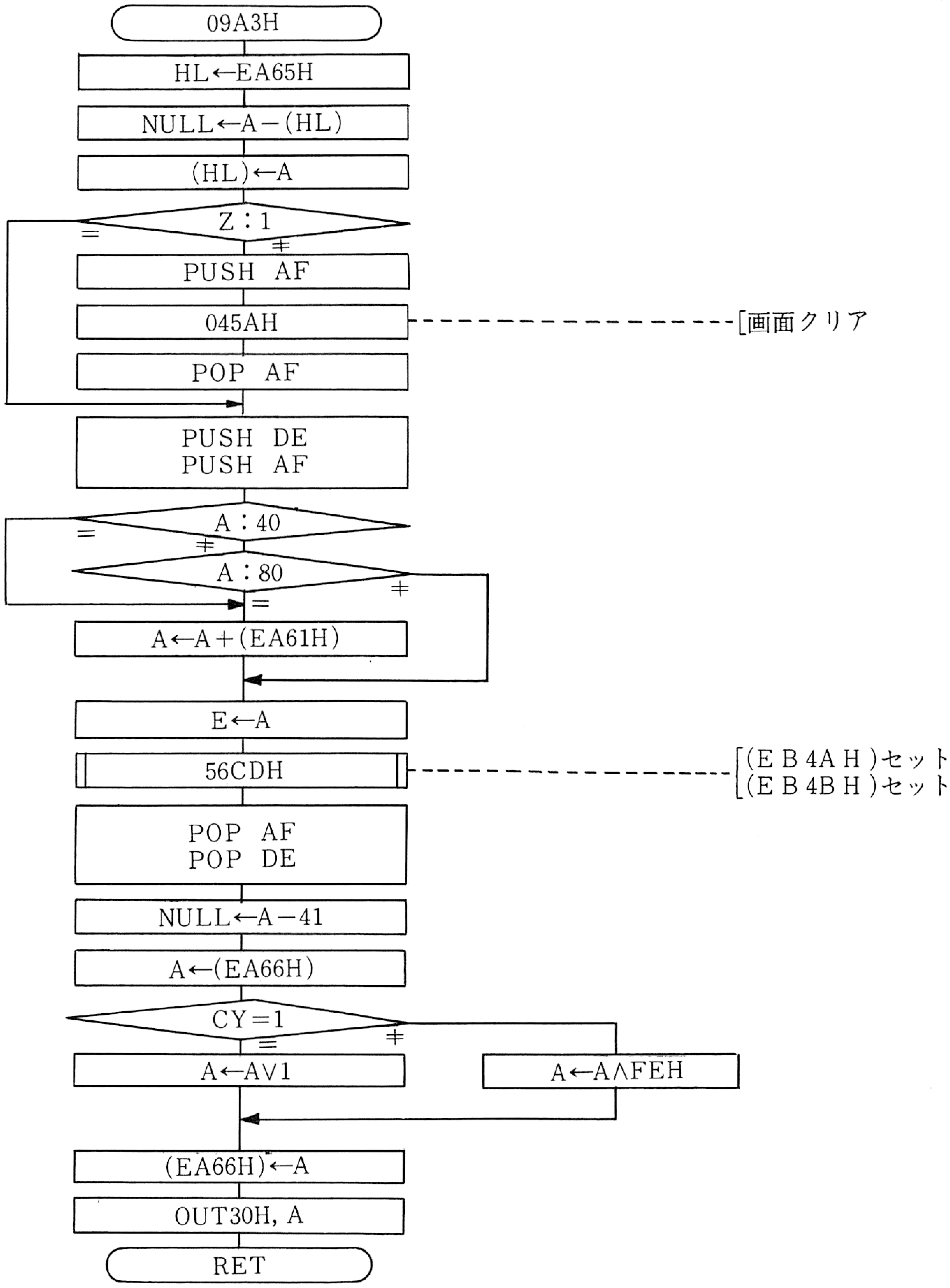
```
LD      A, 80
CALL    09A3H
RET
```

例　４０桁に設定する.

```
LD      A, 40
```

```
CALL  09A3H
RET
```

フロー・
チャート



▶アドレス　０９Ｄ７Ｈ

名称　ＣＲＴ画面表示行数の設定

レジスタ　Ａ，Ｆ，Ｂ，Ｃ，Ｄ，Ｅ，Ｈ，Ｌ

内容　Ａレジスタに行数を入れてコールします．

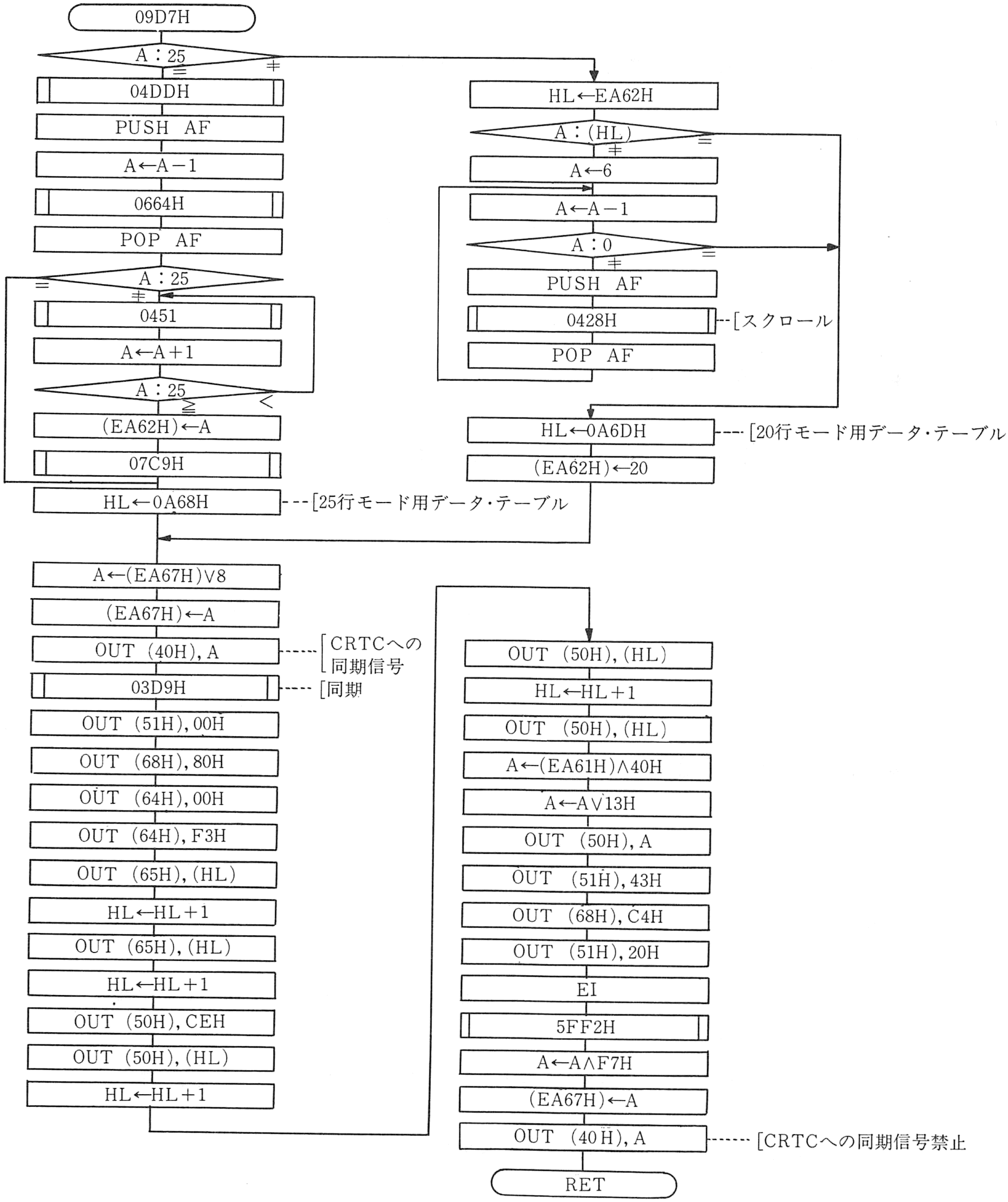
例　２０行に設定する．

```
LD      A, 20
CALL    09D7H
RET
```

例　２５行に設定する．

```
LD      A, 25
CALL    09D7H
RET
```

フロー・
チャート


09D7H
A：25
＝
≠
04DDH
PUSH AF
A←A－1
0664H
POP AF
A：25
0451
A←A＋1
A：25
≧
＜
(EA62H)←A
07C9H
HL←0A68H
[25行モード用データ・テーブル
HL←EA62H
A：(HL)
A←6
A←A－1
A：0
PUSH AF
0428H
[スクロール
POP AF
HL←0A6DH
[20行モード用データ・テーブル
(EA62H)←20
A←(EA67H)∨8
(EA67H)←A
OUT (40H), A
[CRTCへの同期信号
03D9H
[同期
OUT (51H), 00H
OUT (68H), 80H
OUT (64H), 00H
OUT (64H), F3H
OUT (65H), (HL)
HL←HL＋1
OUT (65H), (HL)
HL←HL＋1
OUT (50H), CEH
OUT (50H), (HL)
HL←HL＋1
OUT (50H), (HL)
HL←HL＋1
OUT (50H), (HL)
A←(EA61H)∧40H
A←A∨13H
OUT (50H), A
OUT (51H), 43H
OUT (68H), C4H
OUT (51H), 20H
EI
5FF2H
A←A∧F7H
(EA67H)←A
OUT (40H), A
[CRTCへの同期信号禁止
RET

▶アドレス　０９Ｆ６Ｈ

名称　２５行モードのＣＲＴイニシアライズ

レジスタ　A，F，H，L

内容　ＤＭＡコントローラ及びＣＲＴコントローラを２５行モード用にイニシアライズします．

▶アドレス　０Ａ５ＥＨ

名称　２０行モードのＣＲＴイニシアライズ

レジスタ　A，F，H，L

内容　ＤＭＡコントローラ及びＣＲＴコントローラを２０行モード用にイニシアライズします．

▶アドレス　０Ａ７３Ｈ

名称　ファンクション・キー・リスト

レジスタ　A，F，B，D，E

内容　全てのファンクション・キーに定義されている内容のリストを、ＣＲＴ画面上に表示します．
Ｎ－ＢＡＳＩＣのＫＥＹ　ＬＩＳＴと同様の実行結果を得る事ができます．

例　Ｎ－ＢＡＳＩＣのＫＥＹ　ＬＩＳＴを行う．

```
CALL  0A73H
RET
```

▶アドレス　0B18H

名称　DOSブート

レジスタ　――――

内容　ドライブ1のディスケット上，トラック0セクタ1のデータをPC―8001のRAM上，C000H番地から256バイトにロードし，エラーが発生しなければC000H番地にジャンプします．
エラー発生時には，EDC7H番地をクリアしてもどります．

このサブルーチンは，N―BASICのコールド・スタート時にディスク・ユニットが接続されていれば必ずコールされます．

▶アドレス　0B2EH

名称　CRT画面最下段の消去

レジスタ　A，B，C，D，E，H，L

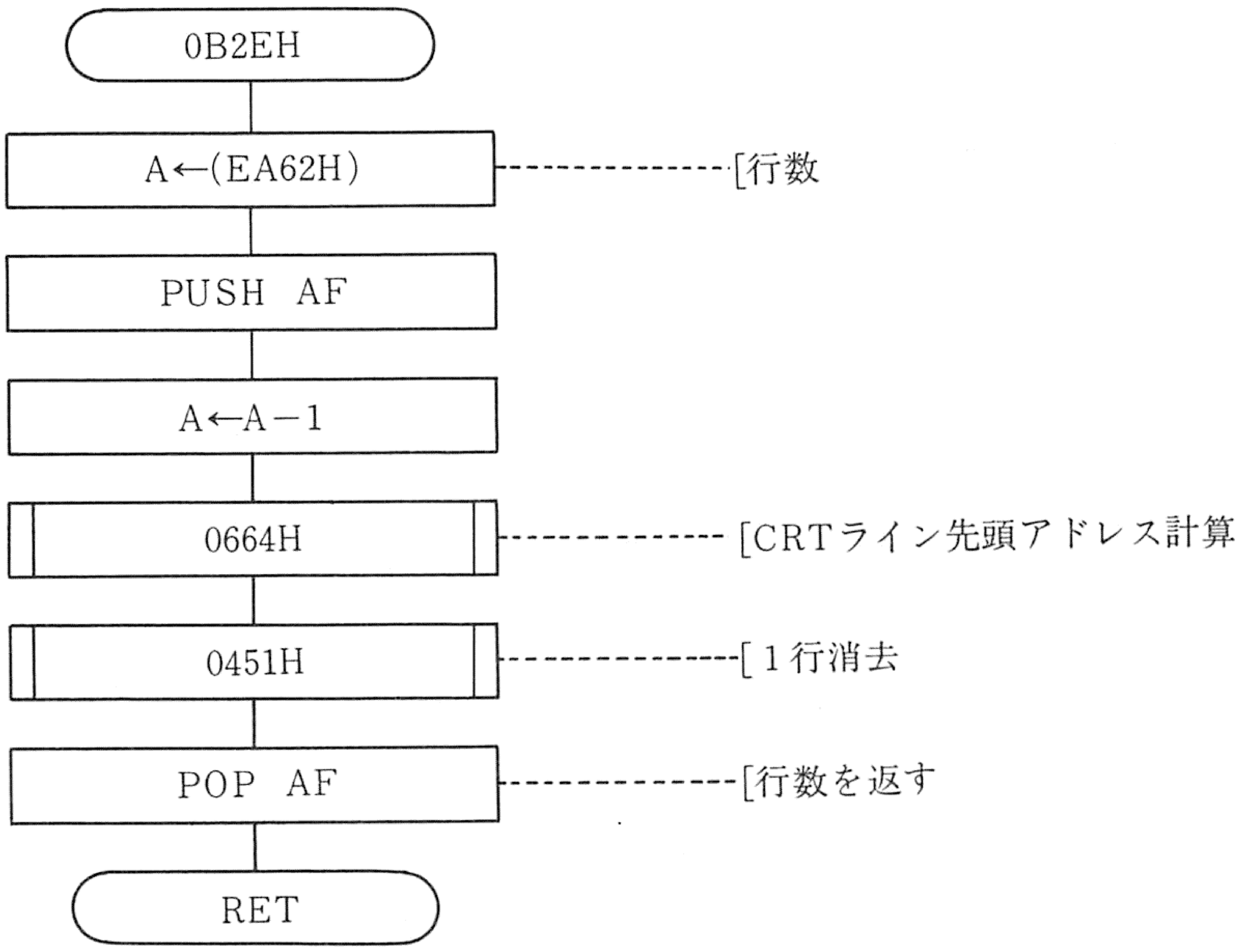
内容　主にファンクション・キー表示等を消去する場合に用います．実際にファンクション・キー表示を行なうか否かはEA60H番地の値が0かFFHかによって決まります．

例　ファンクション・キー表示を止めて，CRT画面上の表示も消去する．

```
XOR     A
LD      (EA60H), A
```

```
CALL   0B2EH
RET
```

フロー・
チャート



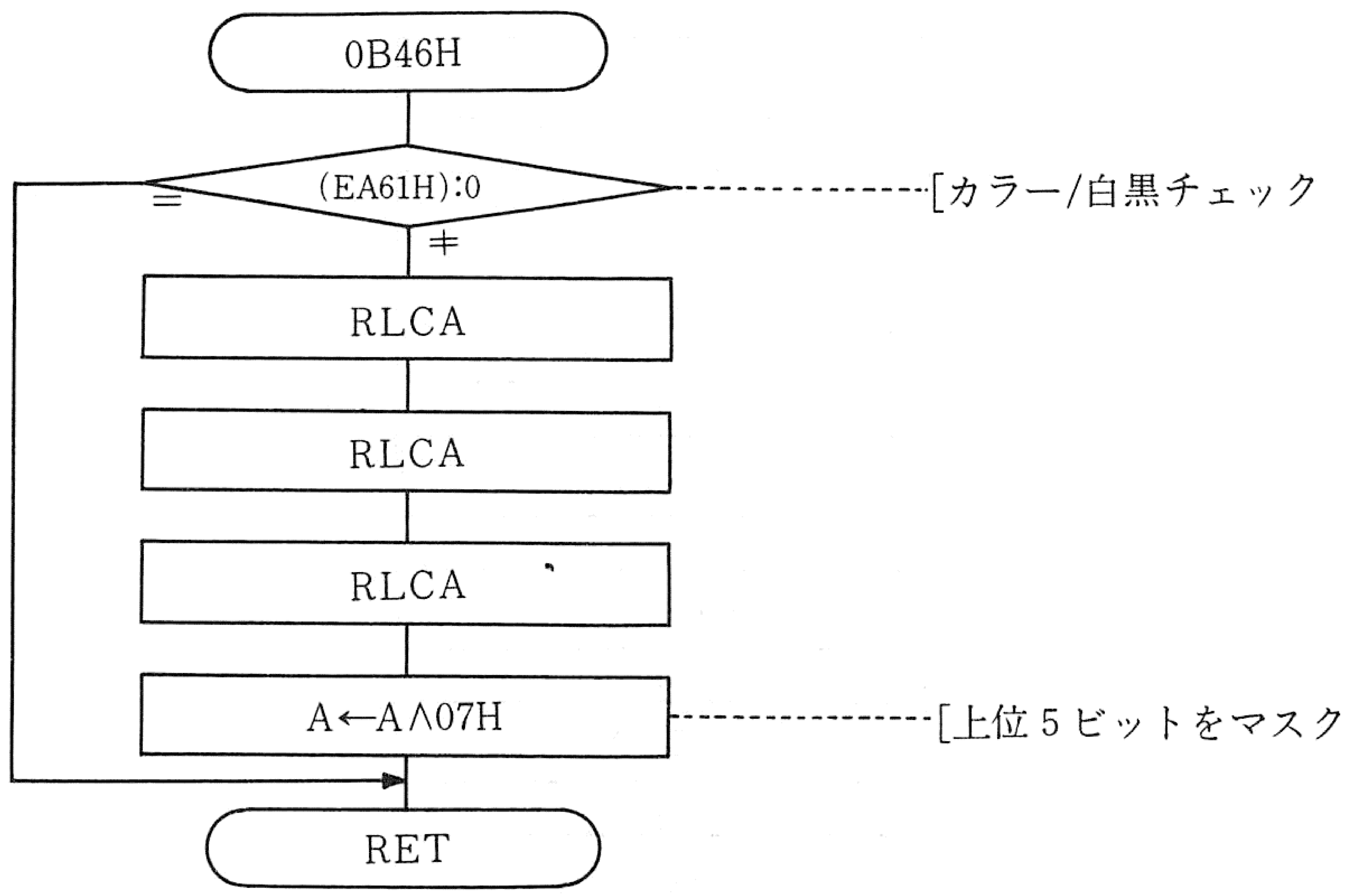
▶アドレス　０Ｂ４６Ｈ

名称　アトリビュート→ファンクション・コード変換

レジスタ　A，F

内容　Aレジスタのアトリビュート・コードをファンクション・コードに変換します．
カラー・モードと白黒モードの判別も，ＥＡ６１Ｈ番地の内容によって自動的に行なわれます．

フロー・
チャート



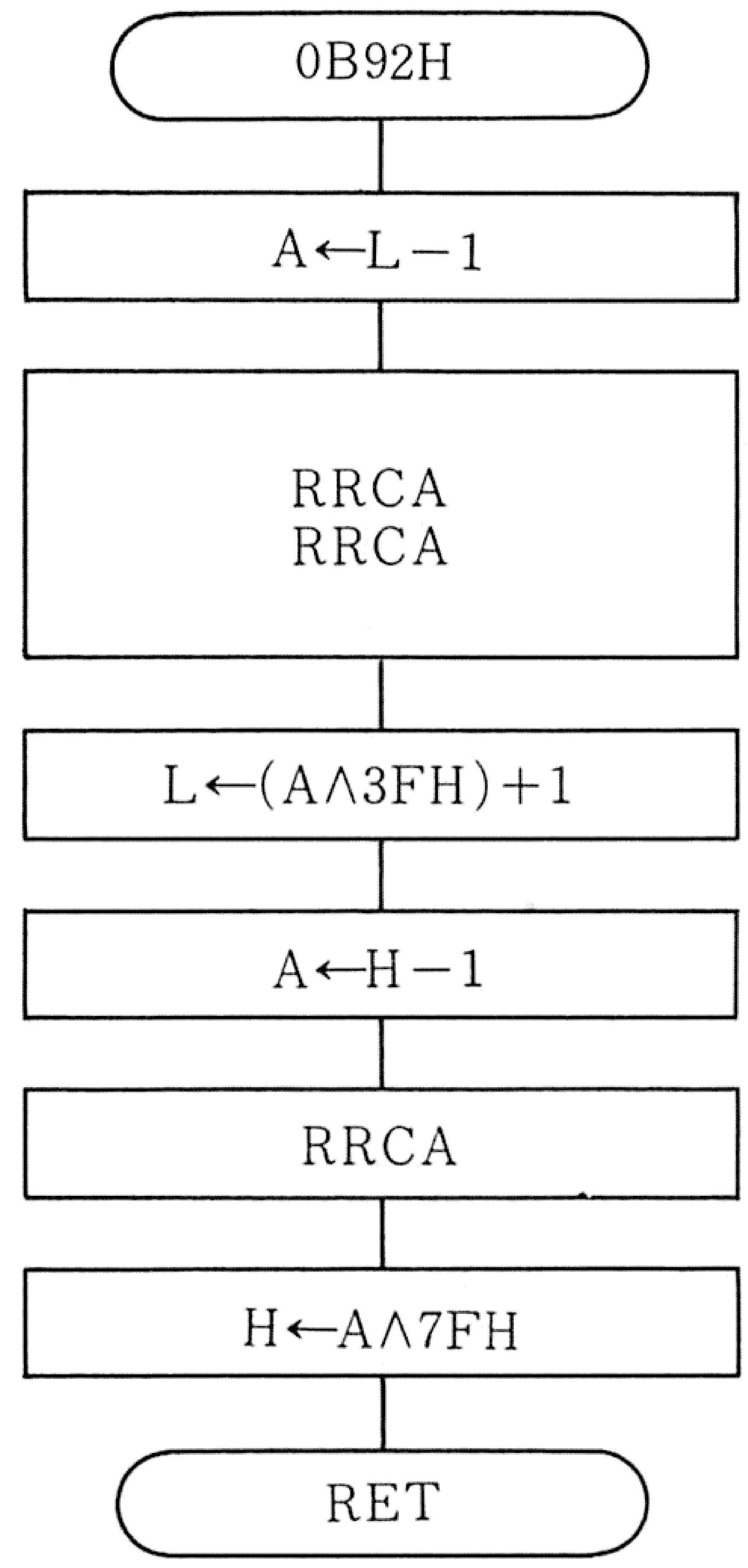
▶アドレス　0B92H

名称　グラフィック座標→キャラクタ座標変換

レジスタ　A，F，H，L

内容　Hレジスタのグラフィック横座標（1～160）とLレジスタのグラフィック縦座標（1～100）を，キャラクタの座標に変換してもどります.

フロー・チャート



▶アドレス　**0BD2H**

名称　カーソル消去

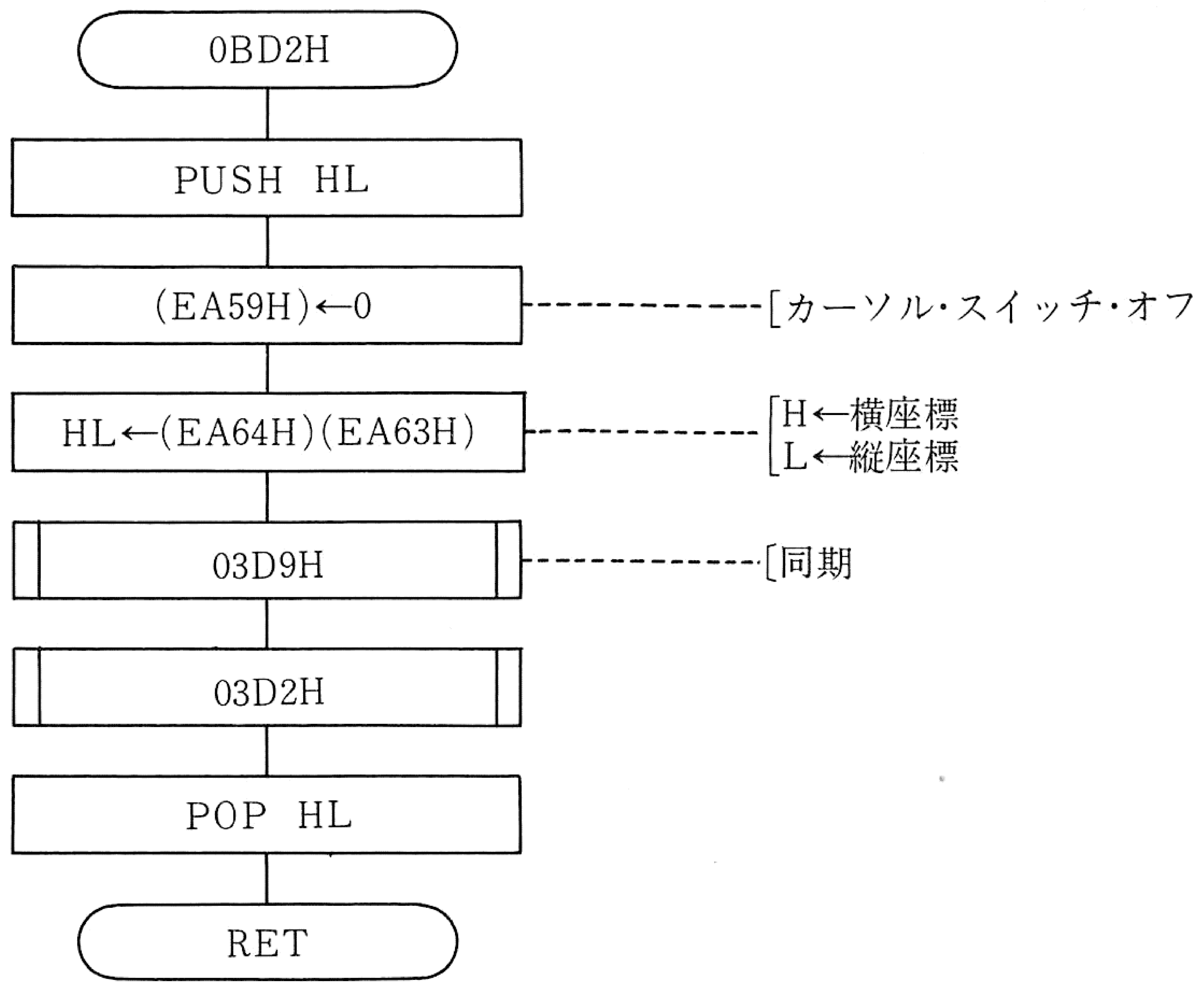
レジスタ　A，F

内容　カーソル位置に表示されているカーソルを消去するためのサブルーチンで，プログラム実行中などにカーソルの点滅がわずらわしい時に用います。N−BASICのLOCATEのカーソル・スイッチと異なり，N−BASICやモニタにもどった場合にはカーソルが表われます．

例　カーソルを消去する．

```
CALL  0BD2H
RET
```

フロー・
チャート



▶アドレス　**０ＢＥ２Ｈ**

名称　**カーソル表示**

レジスタ　**Ａ，Ｆ**

内容　カーソル位置に表示されるべきカーソルを表示するためのサブルーチンで，０ＢＤ２Ｈ番地からのサブルーチンを呼ぶなどして消してあったカーソルを表示させたい場合にコールします．
Ｎ－ＢＡＳＩＣのＩＮＫＥＹ＄を用いている場合にはカーソルが消されていますが，ループ中からこのサブルーチンを呼ぶ事によってカーソルを表示する事が可能となります。

例　カーソルを表示します．

```
CALL  0BE2H
RET
```

▶アドレス　０ＢＦ３Ｈ

名称　ＣＭＴ入力セット

レジスタ　Ａ，Ｆ

内容　８２５１を，ＣＭＴの入力用にイニシアライズし，モーター・リレーをＯＮにして，ＣＭＴのキャリア信号を検出してもどります．
ストップ・キーの入力が有れば，テープ・リード・エラーを出力してＮ－ＢＡＳＩＣに制御を移します．
０ＢＦ３Ｈ番地からＦ１Ｂ６Ｈ番地にフックします．

▶アドレス　０Ｃ２ＥＨ

名称　ＣＭＴクローズ

レジスタ　Ａ，Ｆ

内容　８２５１をクローズし，時間待ちを行なった後に，モーター・リレーをＯＦＦにしてもどります．

▶アドレス　０Ｃ４６Ｈ

名称　ＣＭＴ出力セット

レジスタ　Ａ，Ｆ

内容　８２５１を，ＣＭＴの出力用にイニシアライズし，モーター・リレーをＯＮにして時間待ちを行ないます．
０Ｃ４６Ｈ番地からＦ１Ｂ９Ｈ番地にフックします．

▶アドレス　０Ｃ６９Ｈ

名称　時間待ち

レジスタ　全て保存

内容　Ｄ，Ｅ，Ｈ，Ｌの各レジスタによるループのネスティングによって時間待ちを行ないます．

Ｎ－ＢＡＳＩＣおよびモニタからはＣＭＴなどの時間調整に用いられます．

▶アドレス　０Ｃ８２Ｈ

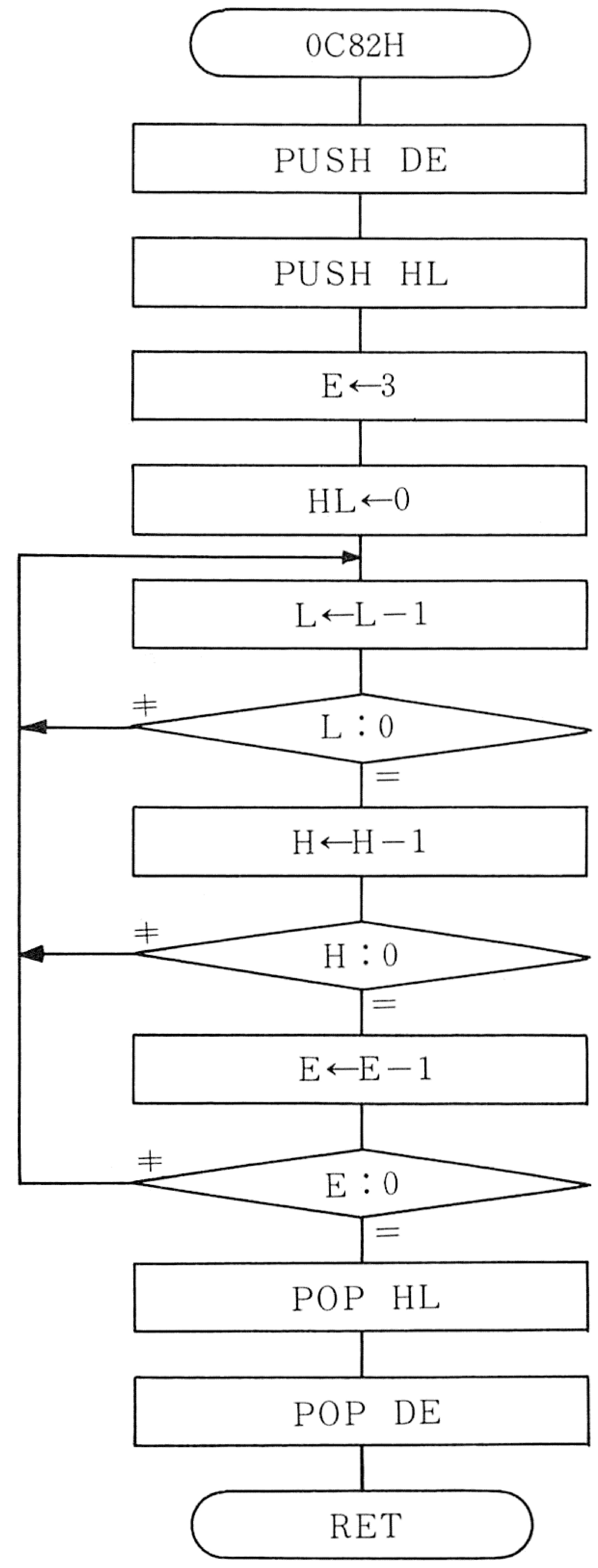
名称　時間待ち

レジスタ　全て保存

内容　Ｄ，Ｅ，Ｈ，Ｌの各レジスタによるループのネスティングによって時間待ちを行ないます．

Ｎ－ＢＡＳＩＣおよびモニタからはＣＭＴなどの時間調整に用いられます．

フロー・
チャート



▶アドレス **０Ｃ８８Ｈ**

名称　ＣＭＴ１バイト入力

レジスタ　Ａ，Ｆ

内容　Ｎ―ＢＡＳＩＣ又は，マシン語モニタからセーブを行なったテープを，ＣＭＴから１バイト入力し，Ａレジスタに入れてもどります．

エラーが発生した場合には，テープ・リード・エラーを出力して，Ｎ―ＢＡ

ＳＩＣに制御を移します．

０Ｃ８８Ｈ番地からはＦ１Ａ４Ｈ番地へのフックを行なっています．

▶アドレス　０ＣＦ１Ｈ

名称　STOPキー，ESCキー・チェック

レジスタ　A，F

内容　STOPキー，ESCキー共に押されていなければ，ゼロ・フラグ（Ｚ）をたてて，STOPキーが押されていれば，キャリー・フラグ（ＣＹ）をたててもどります．

例　ＳＴＯＰキーが押されるまでキー・スキャンを続ける．

```
LBL：CALL  0CF1H
     JR    NC，LBL
     RET
```

▶アドレス　０Ｄ１４Ｈ

名称　８２５１ソフトウェア・リセット

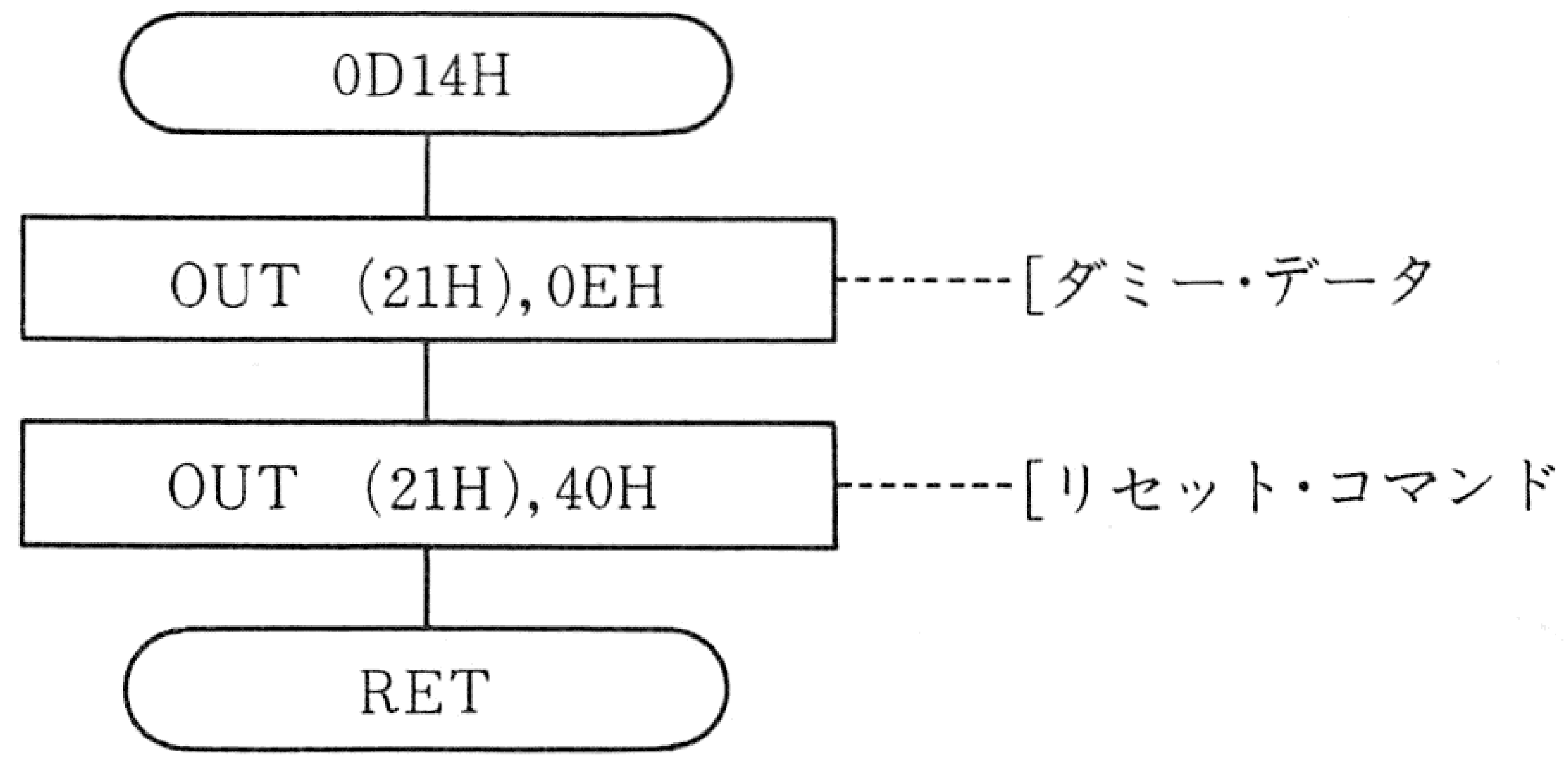
レジスタ　A

内容　８２５１にリセット・コマンドを送って以後モード設定を受け付ける状態にします．

８２５１がリセット直後である場合を考慮して，一度ダミーのデータを出力したあと実際のリセット・コマンドを送っています．

実際には，出力ポート２１Ｈ番地に，０ＥＨと４０Ｈを連続して送っています．

フロー・
チャート



▶アドレス **0D41H**

名称 BEEPエントリ

レジスタ ――――

内容 N－BASICのBEEPエントリ・アドレスです.

▶アドレス **0D43H**

名称 BEEP

レジスタ A

内容 N－BASICのBEEPと同じで一定時間ブザーを鳴らします.
実際には, Aレジスタに7をセットしてRST 18Hを行ない, 1文字表示を行なう事によってブザーを鳴らしています.

例 N－BASICのBEEPを行なう.

```
CALL 0D43H
RET
```

▶アドレス　０Ｄ４ＢＨ

名称　ＢＥＥＰｎ

レジスタ　Ａ，Ｆ，Ｅ

内容　Ａレジスタの値が０の時にブザーを鳴り止め，０以外の場合にはブザーを鳴らします．

例　ブザーを鳴り止める．

```
XOR    A
CALL   0D4BH
RET
```

例　ブザーを鳴らす．

```
LD     A, 1
CALL   0D4BH
RET
```

▶アドレス　０Ｄ４ＥＨ

名称　ＢＥＥＰ１

レジスタ　Ａ，Ｆ，Ｅ

内容　Ｎ－ＢＡＳＩＣのＢＥＥＰ１と同じで内蔵スピーカを鳴らします．
Ｎ－ＢＡＳＩＣでは出力ポート４０Ｈの第５ビットを直接制御する事によってスピーカを制御する事が可能ですが，出力ポート４０Ｈには他にも幾つかの周辺機器が接続されていますので，他のビットが変化しない様にＥＡ６７Ｈ番地の値と論理和（ＯＲ）をとる等の注意が必要です．
もちろんシステム・サブルーチンを利用する場合には，それらの処理は自動

的に行なわれます.

▶アドレス　**０Ｄ５ＤＨ**

名称　**ＭＯＮエントリ**

レジスタ　―――――

内容　Ｎ－ＢＡＳＩＣのＭＯＮエントリ・アドレスで，マシン語モニタの先頭アドレスである５Ｃ２ＣＨ番地へジャンプします.

▶アドレス　**０Ｄ６０Ｈ**

名称　**プリンタへの１バイト出力**

レジスタ　**全て保存**

内容　Ａレジスタのデータ１バイトをプリンタへ出力します.
出力以前にＦ１ＢＣＨ番地をコールしています.

例　Ｎ－ＢＡＳＩＣコールド・スタート時のタイトル・メッセージをプリントする.

```
        LD    HL, 1838H
LBL:    LD    A, (HL)
        CP    0
        RET   Z
        CALL  0D60H
        INC   HL
        JR    LBL
```

▶アドレス　0DA1H

名称　MOTORエントリ

レジスタ　――――

内容　N－BASICのMOTORエントリ・アドレスです．

▶アドレス　0DA3H

名称　MOTOR

レジスタ　A，F

内容　内蔵リレーの状態を反転します．
内蔵リレーは出力ポート30H番地の第3ビットに接続されていますが，スピーカの制御と同様に他のビットを変化させない様に注意する心要があります．30Hに出力したデータはEA66H番地に保存されていますので参照すれば良いでしょう．

例　内蔵リレーの状態を反転する．

```
CALL  0DA3H
RET
```

▶アドレス　0DAEH

名称　MOTER n

レジスタ　A，F

内容　Ａレジスタの値が０の時にＭＯＴＯＲ０を行ない，０以外の場合にはＭＯＴＯＲ１を行ないます．

例　ＭＯＴＯＲ０を行なう．

```
XOR   A
CALL  0DAEH
RET
```

例　ＭＯＴＯＲ１を行なう．

```
LD    A, 1
CALL  0DAEH
RET
```

▶アドレス　０ＤＢ８Ｈ

名称　ＴＥＲＭエントリ

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのＴＥＲＭエントリ・アドレスです．

▶アドレス　０Ｆ７５Ｈ

名称　１文字入力待ち

レジスタ　Ａ，Ｆ

内容　キー入力が有るまで，０Ｆ７ＢＨからのキー入力サブルーチンをコールし続け，入力したキーのキャラクタ・コードをＡレジスタに入れてもどります．ＣＲＴ画面へのエコー・バックは行ないません．

例　1文字キー入力を行ない同時にＣＲＴ画面へもエコー・バックする.

```
CALL  0F75H
CALL  0257H
RET
```

▶アドレス　０Ｆ７ＢＨ

名称　1文字入力

レジスタ　A, F

内容　キー入力が有ればキャリー・フラグ（ＣＹ）をたおし，Ａレジスタにキャラクタ・コードを入れてもどります.
キー入力が無い場合にはキャリー・フラグ（ＣＹ）を立て，Ａレジスタには0を入れてもどります.
キー入力，特に特殊キーの入力が有ると，処理速度がいちじるしく落ちるためリアル・タイムな処理の場合には，直接入力ポートをスキャンした方が良い様です.

▶アドレス　０ＦＡＣＨ

名称　キー・スキャニング

レジスタ　A, F

内容　カーソルを表示しないで，キーボードをスキャンし，Ａレジスタにキャラクタ・コードを入れてもどります．キー入力が無い場合にはキャリー・フラグ（ＣＹ）をセットしてもどります.
ただし，ＣＲＴ画面へのエコー・バックは行ないません.

▶アドレス　１０Ｅ５Ｈ～１１７ＣＨ　（ＤＡＴＡ）

名称　キー・マトリクス→アスキー・コード変換データ

レジスタ　――――

内容　キー・センス用の入力ポートから入力したキーボード・マトリクスの入力データをアスキー・コードに変換するためのデータです.

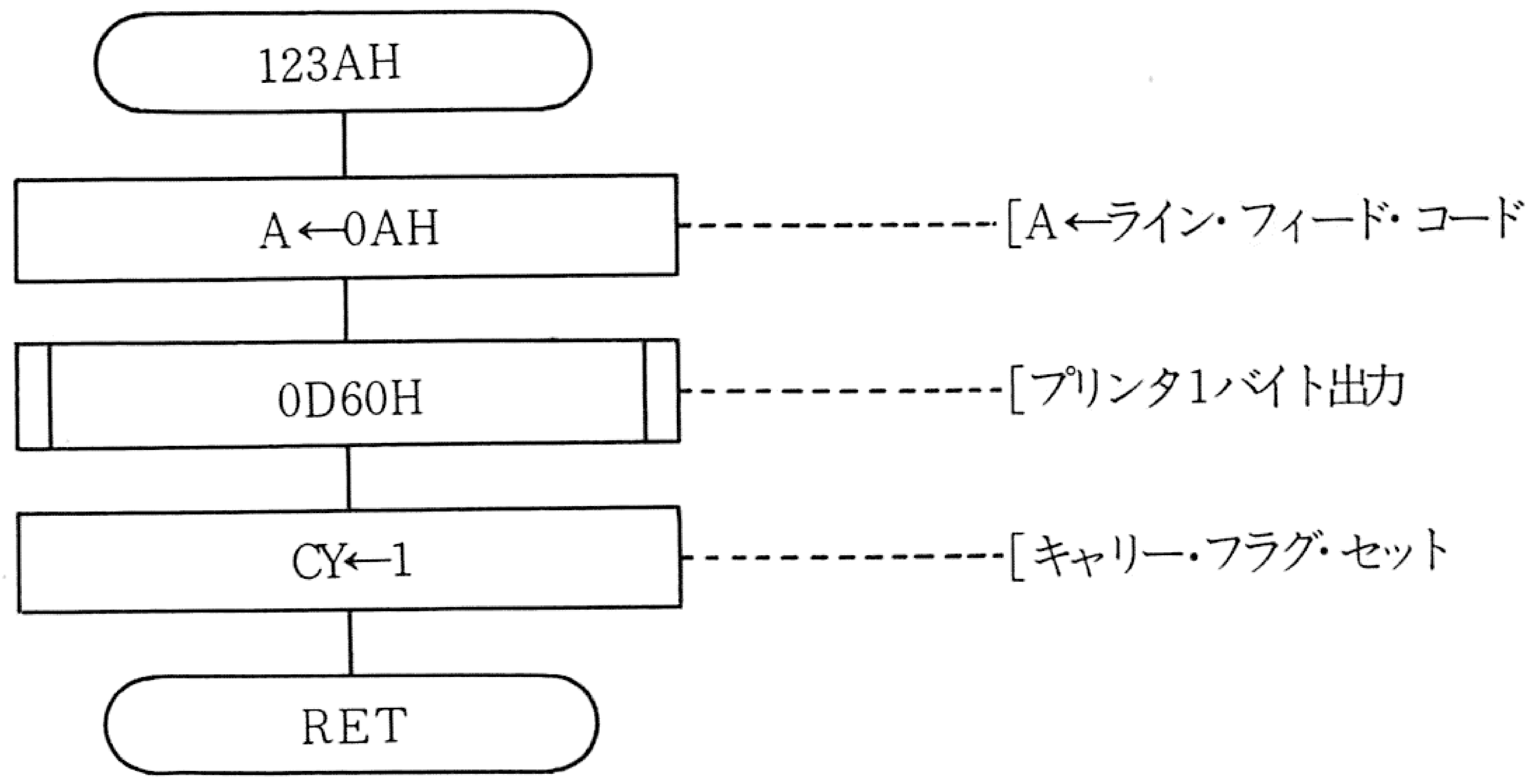
▶アドレス　１２３ＡＨ

名称　プリンタ改行

レジスタ　Ａ，Ｆ

内容　プリンタに，ライン・フィード・コード（０ＡＨ）を出力します.

フロー・チャート



▶アドレス　１２４ＡＨ

名称　**画面コピー**

レジスタ　A，F，B，C，D，E，H，L

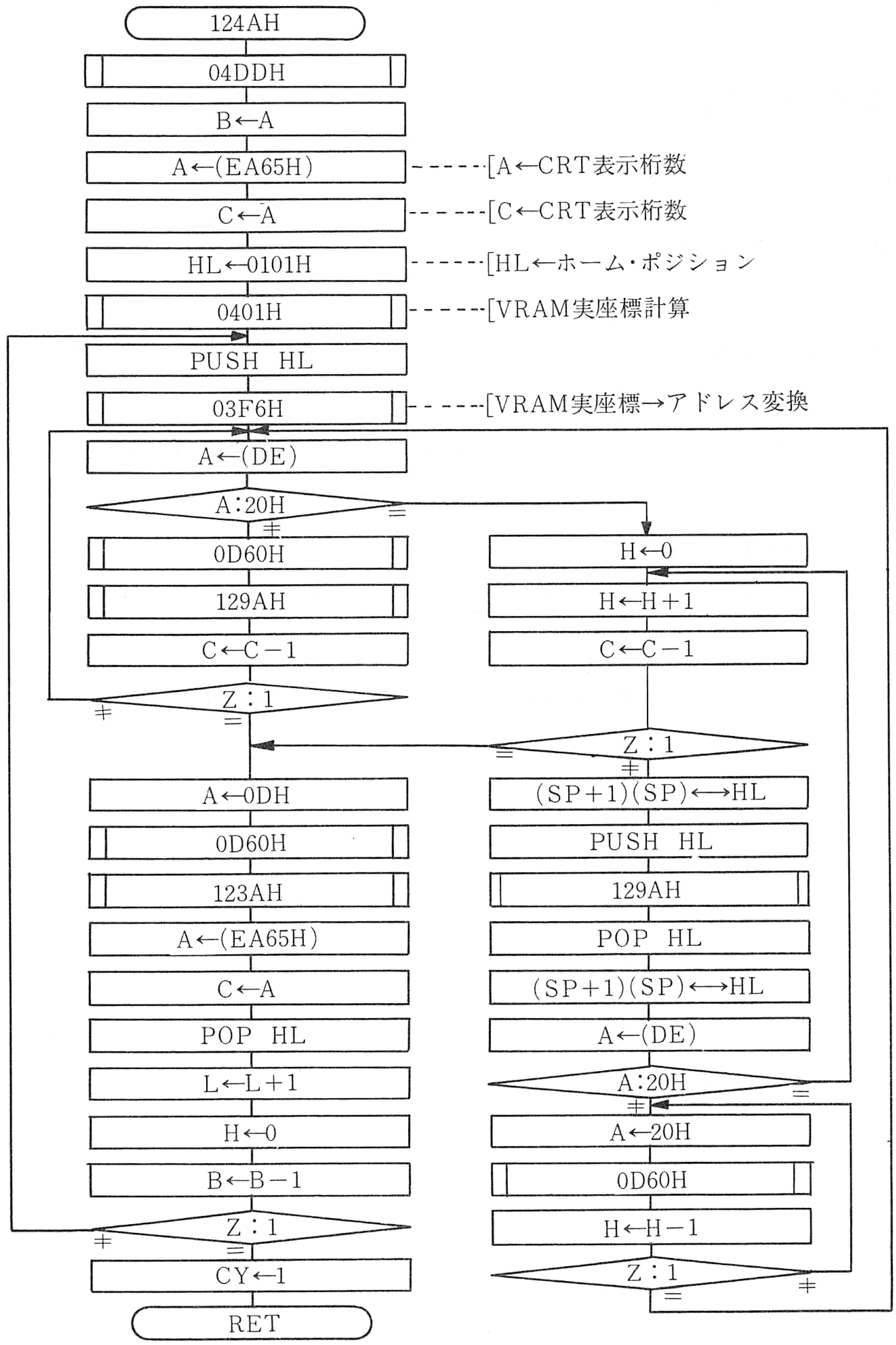
内容　ＣＲＴ画面に表示されているキャラクタのイメージをそのままプリンタへ出力します．

ただし，ヌル・キャラクタ・コードは２０Ｈである心要があり，カラー指定などのアトリビュート特性は無視されるため，ドット・グラフィックによる画面は，正常にコピーされません．

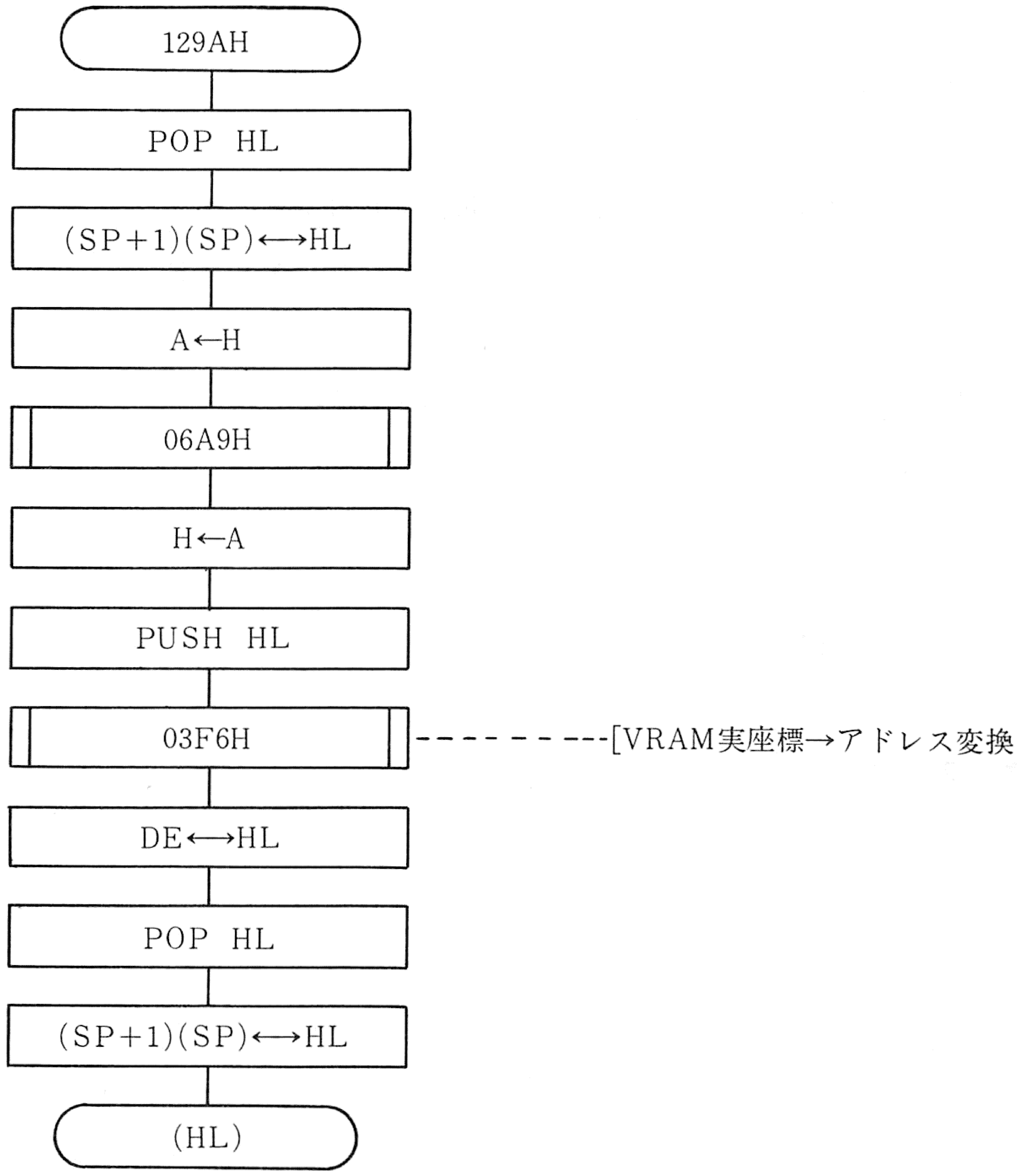
出力は，ＣＲＴ画面の最上段から，スクロール範囲の下端まで行なわれます．

また，ＥＡ５８Ｈ番地の第０ビットを立てておき，［f・9］キーによっても同様の画面コピーを行なう事ができます．

フロー・
チャート



124AH
04DDH
B←A
A←(EA65H)
[A←CRT表示桁数
C←A
[C←CRT表示桁数
HL←0101H
[HL←ホーム・ポジション
0401H
[VRAM実座標計算
PUSH HL
03F6H
[VRAM実座標→アドレス変換
A←(DE)
A:20H
0D60H
129AH
C←C−1
Z:1
A←0DH
0D60H
123AH
A←(EA65H)
C←A
POP HL
L←L+1
H←0
B←B−1
Z:1
CY←1
RET
H←0
H←H+1
C←C−1
Z:1
(SP+1)(SP)←→HL
PUSH HL
129AH
POP HL
(SP+1)(SP)←→HL
A←(DE)
A:20H
A←20H
0D60H
H←H−1
Z:1



▶アドレス　１３４３H

名称　KEYエントリ

レジスタ　――――

内容　N―BASICのKEYエントリ・アドレスです.

▶アドレス　１６０２H

名称　タイマ・リード

レジスタ　A, F, B, C, D, E, H, L

内容　タイマＩＣの値を，ＥＡ７６Ｈ～ＥＡ７ＢＨ番地のタイマ制御用ワーク・エリアにＢＣＤコードで読み込みます．ただし，ＹＥＡＲ（年）の値は読み込みません．

ＨＬレジスタ対には，ＥＡ７ＢＨが入ります．

詳しくは，本文を参照してください．

▶アドレス　１６４ＣＨ

名称　タイマ・クリア

レジスタ　A, F, B, C, D, E, H, L

内容　タイマＩＣに初期値をセットしてもどります．

このサブルーチンをコールすると，タイマＩＣ，及び，ＥＡ７６Ｈ～ＥＡ７ＢＨ番地のワーク・エリアには，７９年０１月０１日の００時００分００秒がセットされます．

詳しくは，本文を参照してください．

なお，００６ＡＨ番地のＮ－ＢＡＳＩＣホット・スタートでは，まっ先にこのサブルーチンをコールしています．

▶アドレス　１６６３Ｈ

名称　タイマ・ライト

レジスタ　A, F, B, C, D, E, H, L

内容　タイマ制御用のワーク・エリアである，ＥＡ７６Ｈ～ＥＡ７Ｂ番地のＢＣＤコードをタイマＩＣに書き込みます．ただし，ＹＥＡＲ（年）の値はＩＣに

設定しません．

ＨＬレジスタ対には，ＥＡ７ＢＨが入ります．

詳しくは，本文を参照してください．

▶アドレス　**１７５７Ｈ**

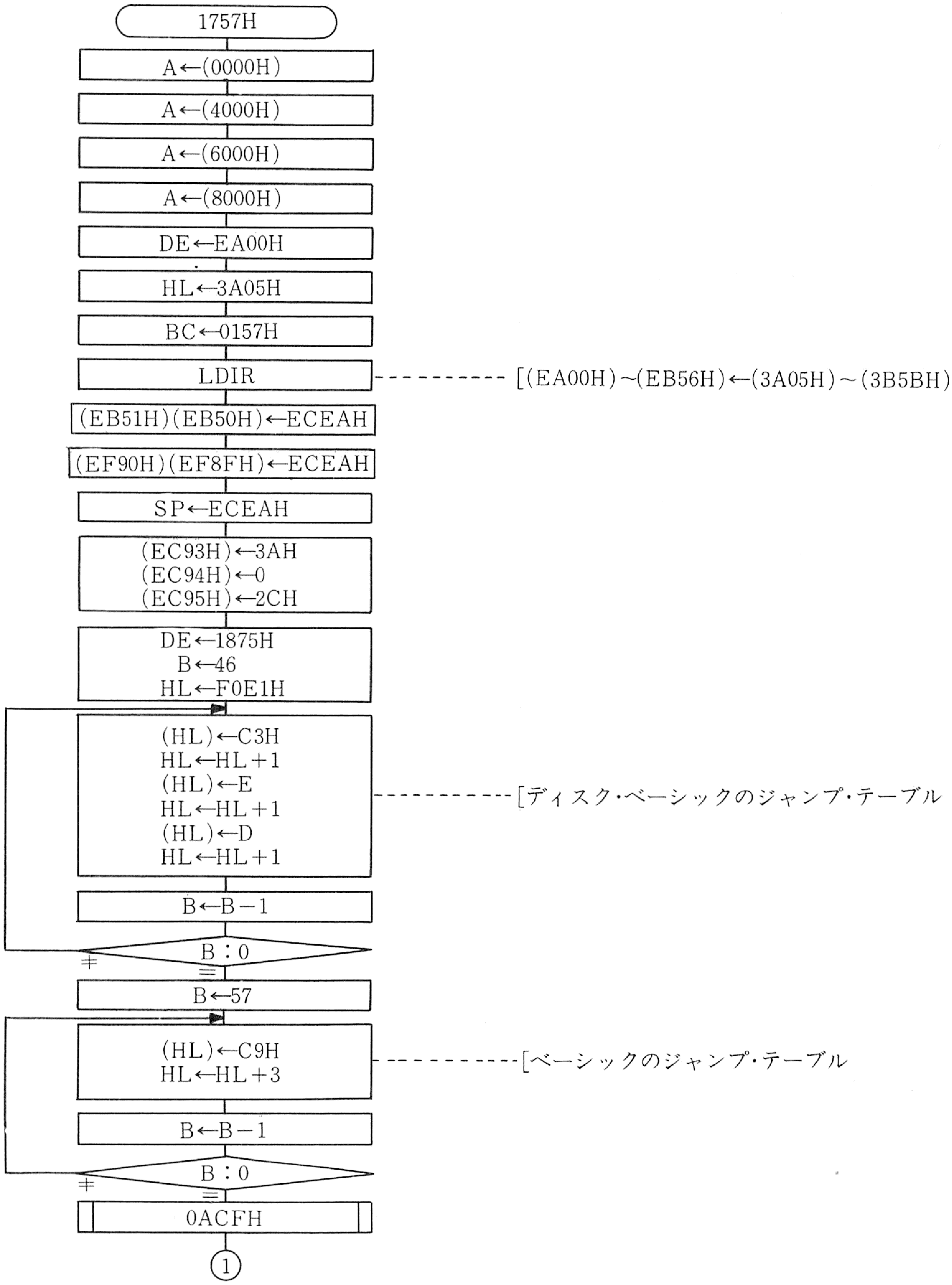
名称　**Ｎ－ＢＡＳＩＣイニシアライズ**

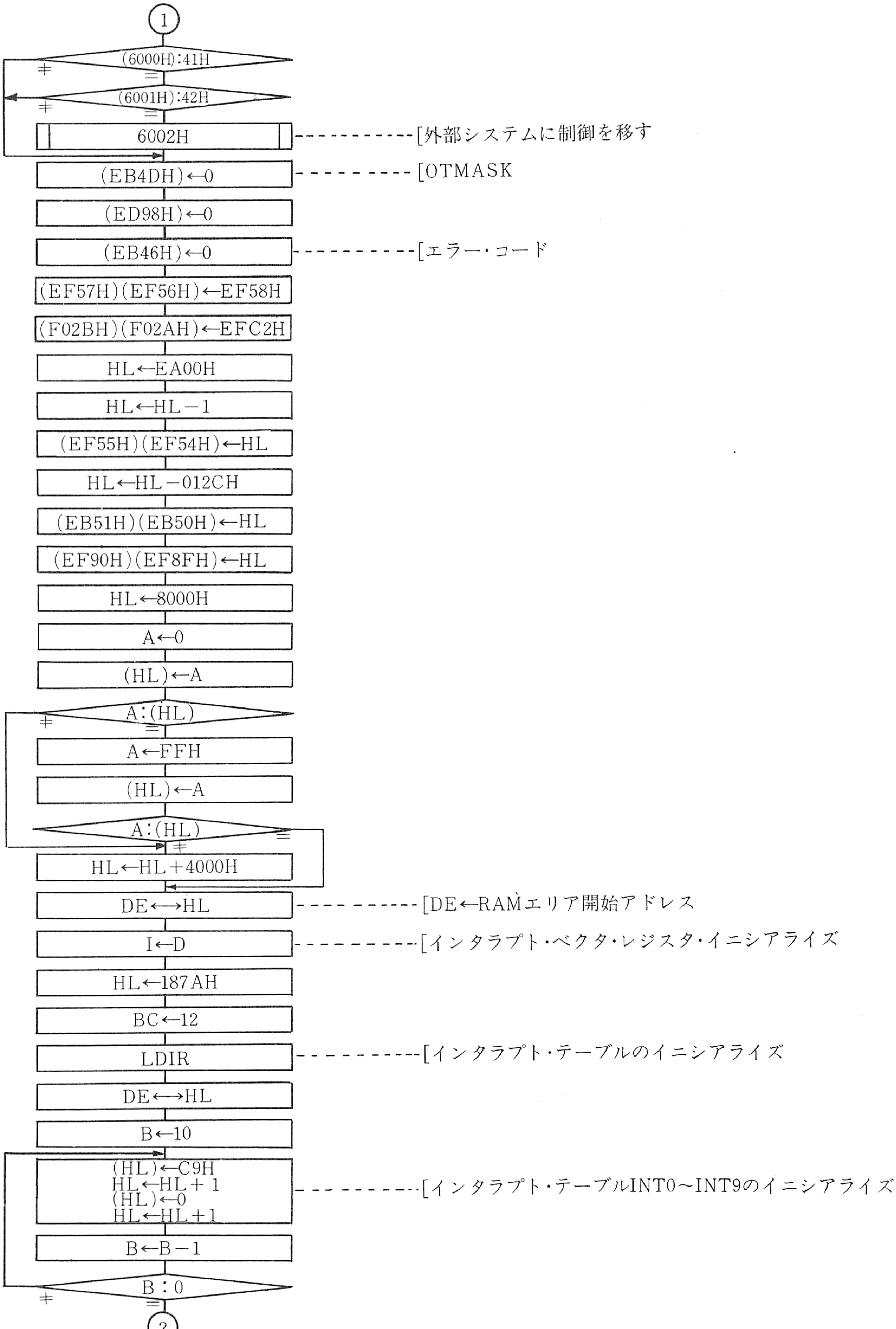
レジスタ　————

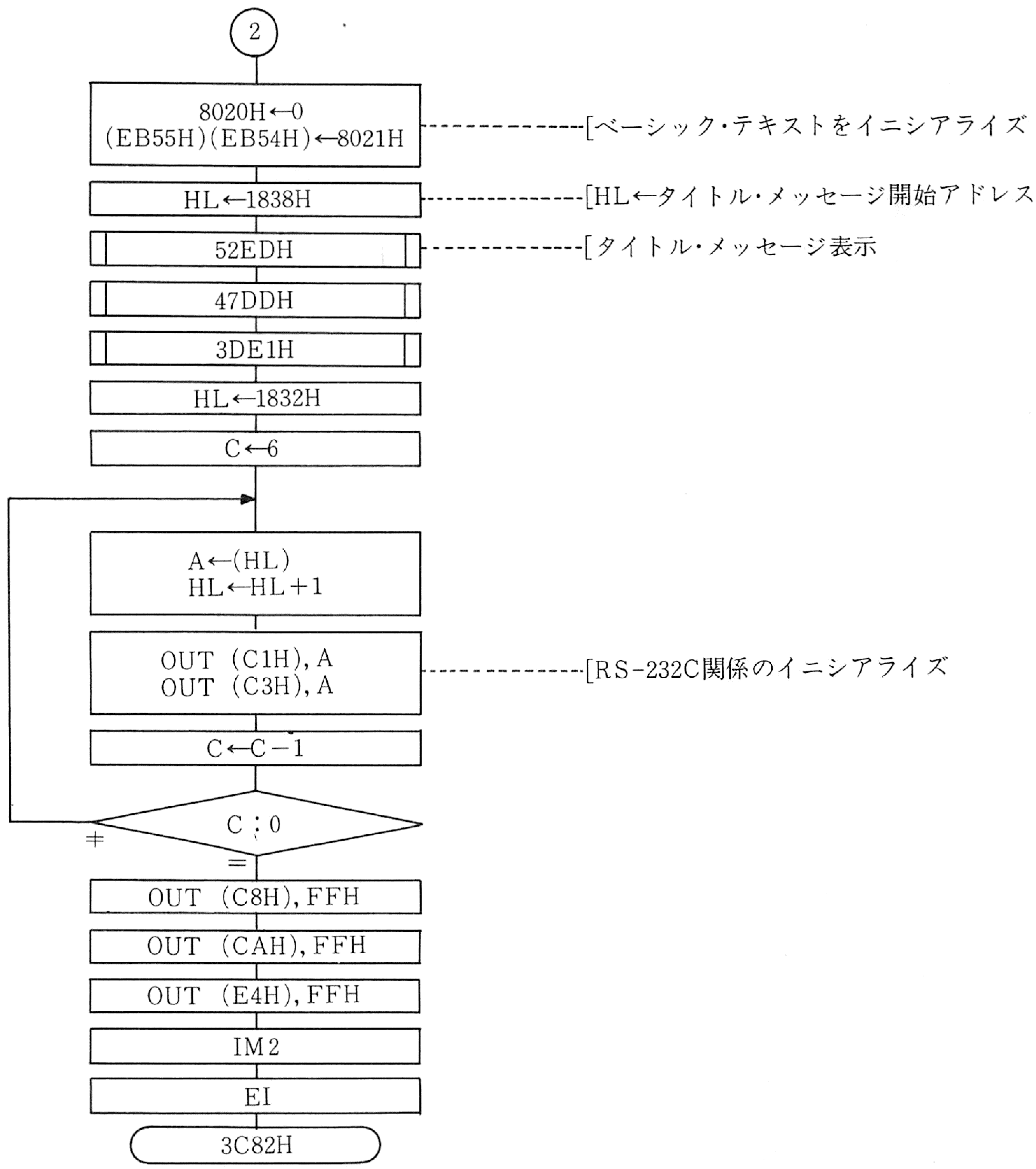
内容　Ｎ－ＢＡＳＩＣのイニシアライズ，テキストのクリアなどを行ない，Ｎ－ＢＡＳＩＣに制御を移します．

ＰＣ－８００１のリセット時にストップ・キーが押されていない場合に，コールド・スタートとしてジャンプして来るルーチンです．

フロー・チャート


1757H
A←(0000H)
A←(4000H)
A←(6000H)
A←(8000H)
DE←EA00H
HL←3A05H
BC←0157H
LDIR
[(EA00H)~(EB56H)←(3A05H)~(3B5BH)
(EB51H)(EB50H)←ECEAH
(EF90H)(EF8FH)←ECEAH
SP←ECEAH
(EC93H)←3AH
(EC94H)←0
(EC95H)←2CH
DE←1875H
B←46
HL←F0E1H
(HL)←C3H
HL←HL+1
(HL)←E
HL←HL+1
(HL)←D
HL←HL+1
[ディスク・ベーシックのジャンプ・テーブル
B←B−1
B：0
≠
＝
B←57
(HL)←C9H
HL←HL+3
[ベーシックのジャンプ・テーブル
B←B−1
B：0
≠
＝
0ACFH
1


(6000H):41H
(6001H):42H
6002H
[外部システムに制御を移す
(EB4DH)←0
[OTMASK
(ED98H)←0
(EB46H)←0
[エラー・コード
(EF57H)(EF56H)←EF58H
(F02BH)(F02AH)←EFC2H
HL←EA00H
HL←HL−1
(EF55H)(EF54H)←HL
HL←HL−012CH
(EB51H)(EB50H)←HL
(EF90H)(EF8FH)←HL
HL←8000H
A←0
(HL)←A
A:(HL)
A←FFH
(HL)←A
A:(HL)
HL←HL+4000H
DE←→HL
[DE←RAMエリア開始アドレス
I←D
[インタラプト・ベクタ・レジスタ・イニシアライズ
HL←187AH
BC←12
LDIR
[インタラプト・テーブルのイニシアライズ
DE←→HL
B←10
(HL)←C9H
HL←HL+1
(HL)←0
HL←HL+1
[インタラプト・テーブルINT0〜INT9のイニシアライズ
B←B−1
B:0



▶アドレス　１７Ｅ９Ｈ

名称　Ｎ－ＢＡＳＩＣテキスト先頭アドレス設定

レジスタ　――――

内容　ＨＬレジスタ対で指定するアドレスがＮ－ＢＡＳＩＣテキストの先頭アドレスに設定されて，Ｎ－ＢＡＳＩＣをスタートします．

以後入力されるN－BASICのテキスト・プログラムは，指定したアドレスから収納されて行きます．

例　64KバイトRAMのシステムで，N－BASICのテキスト先頭アドレスを6000H番地に設定する．

```
LD    HL，6000H
JP    17E9H
```

▶アドレス　1838H～1874H　（DATA）

名称　N－BAS1Cタイトル・メッセージ

レジスタ　――――

内容　N－BAS1CのコールドＦ・スタート時にCRT画面に表示されるメッセージで，末尾に0を伴います．

▶アドレス　1875H

名称　Disk　BASIC　Featureエラー出力

レジスタ　――――

内容　Disk　BASIC　Featureのエラーを出力して，N－BASICに制御を移します．

Disk－BASICを起動させていない場合には，Disk－BASICのジャンプ・テーブルを介して，このアドレスにジャンプします．

▶アドレス　1886H

名称　ＧＥＴエントリ

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのＧＥＴエントリ・アドレスです．

▶アドレス　１８９１Ｈ

名称　ＰＵＴエントリ

レジタス　――――

内容　Ｎ－ＢＡＳＩＣのＰＵＴエントリ・アドレスです．

▶アドレス　１Ｂ７ＥＨ

名称　スクリーン・エディタ

レジスタ　A，F，B，C，D，E，H，L

内容　スクリーン・エディトを行ない入力データを，ＥＣ９６Ｈ～ＥＤ９５Ｈ番地の入力バッファに入れてもどります．
リターン・キーの時キャリー・フラグ（ＣＹ）をたおし，ストップ・キーでキャリー・フラグ（ＣＹ）をたてます．
入力データの最後には，0が入り，ＨＬレジスタ対にはＥＣ９５Ｈが入ります．

例　スクリーン・エディトで入力した１行分のデータをＣＲＴ画面に表示する．

```
CALL  1B7EH
CALL  52ECH
```

```
        RET
```

▶アドレス **1B8AH**

名称 1行入力

レジスタ A, F, B, C, D, E, H, L

内容 EC96H〜ED95H番地の入力バッファに1行分のデータをキーボードから入力し，入力データの最後には0を，HLレジスタ対にはEC95Hを入れてもどります.

例 入力バッファへ1行入力し入力データ中の全英小文字を英大文字に変換する.

```
        CALL  1B8AH
LBL:    INC   HL
        LD    A, (HL)
        CP    0
        RET   Z
        CALL  5FC1H
        LD    (HL), A
        JR    LBL
```

▶アドレス **1EC0H**

名称 CSAVEエントリ

レジスタ ――――

内容 N―BASICのCSAVEエントリ・アドレスです.

▶アドレス　１Ｆ１０

名称　ＣＬＯＡＤエントリ

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのＣＬＯＡＤエントリ・アドレスです．

▶アドレス　１ＦＣ９Ｈ～１ＦＣＥＨ　（ＤＡＴＡ）

名称　Ｂａｄ$^{C}{}_{R}{}^{L}{}_{F}$メッセージ

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのＣＬＯＡＤ？でエラーが発生した場合に出力される．Ｂａｄ$^{C}{}_{R}{}^{L}{}_{F}$メッセージで末尾に０を伴います．

▶アドレス　１ＦＣＦＨ

名称　ファイル・ネーム収納

レジスタ　Ａ，Ｆ，Ｂ，Ｃ，Ｄ，Ｅ，Ｈ，Ｌ

内容　ＨＬレジスタ対で指定するアドレスから収納されている，ダブル・クォーテーション・マーク「”」でくくられた文字列を，ファイル・ネームとして，ＥＦ３６Ｈ～ＥＦ３ＢＨ番地のサーチ・バッファに収納してもどります．
文字列が６文字に満たない場合には，余った領域を０で埋めます．

▶アドレス　１ＦＥＦＨ～１ＦＦ５Ｈ　（ＤＡＴＡ）

名称　Ｆｏｕｎｄ：メッセージ

レジスタ　――――――

内容　Ｎ―ＢＡＳＩＣで，ＣＭＴからのプログラム・ロード時に出力する，Ｆｏｕｎｄ：メッセージで末尾に0を伴います．

▶アドレス　１ＦＦ６Ｈ～１ＦＦＢＨ　（ＤＡＴＡ）

名称　Ｓｋｉｐ：メッセージ

レジスタ　――――――

内容　Ｎ―ＢＡＳＩＣで，ＣＭＴからのプログラム・スキップ時に出力される，Ｓｋｉｐ：メッセージで末尾に0を伴います．

▶アドレス　１ＦＦＣＨ

名称　ファイル・ネーム表示

レジスタ　B, C, H, L

内容　ＨＬレジスタ対で示すアドレスからの0が収納されているアドレスまでの文字列を出力した後，ＥＦ３ＥＨ番地から6バイトに渡るファイル・ネーム・バッファのファイル・ネームを出力します．
文字列「Ｆｏｕｎｄ：」は１ＦＥＦＨ番地から，「Ｓｋｉｐ：」は１ＦＦ６Ｈ番地からそれぞれキャラクター・コードで収納されています．

▶アドレス　２０Ｆ９Ｈ

名称　ＰＯＲＴエントリ

レジスタ　――――

内容　Ｎ―ＢＡＳＩＣのＰＯＲＴエントリ・アドレスです.

▶アドレス　２２６２Ｈ

名称　ＩＮＩＴエントリ

レジスタ　――――

内容　Ｎ―ＢＡＳＩＣのＩＮＩＴエントリ・アドレスです.

▶アドレス　２４０ＦＨ

名称　単精度実数型数値の減算

レジスタ　Ａ，Ｂ，Ｃ，Ｄ，Ｅ，Ｈ，Ｌ

内容　ＥＤＣＢレジスタ内の単精度実数型数値から，フローティング・アキュームレータ内の単精度実数型数値を減じた差を求め，フローティング・アキュームレータに格納します.

▶アドレス　２４１２Ｈ

名称　単精度実数型数値の加算

レジスタ　Ａ，Ｆ，Ｂ，Ｃ，Ｄ，Ｅ，Ｈ，Ｌ

内容　ＥＤＣＢレジスタ内の単精度実数型数値にフローティング・アキュームレータ内の単精度実数型数値を加えた和をフローティング・アキュームレータに格納します.

▶アドレス　２５０３Ｈ

名称　ＬＯＧエントリ

レジスタ　A, F, B, C, D, E, H, L

内容　フローティング・アキュームレータの単精度実数の自然対数をもとめ，フローティング・アキュームレータに格納します.

▶アドレス　２５４１Ｈ

名称　単精度実数型数値の乗算

レジスタ　A, F, B, C, D, E, H, L

内容　ＥＤＣＢレジスタ内の単精度実数型数値に，フローティング・アキュームレータ内の単精度実数型数値を乗じた積を，フローティング・アキュームレータに格納します.

▶アドレス　２５９ＣＨ

名称　単精度実数型数値の除算

レジスタ　A, F, B, C, D, E, H, L

内容　ＥＤＣＢレジスタ内の単精度実数型数値をフローティング・アキュームレータ内の単精度実数型数値で割った商を，フローティング・アキュームレータに格納します．

▶アドレス　２６７１Ｈ

名称　ＡＢＳエントリ

レジスタ　A，B，C，D，E，H，L

内容　フローティング・アキュームレータ内の数値データの絶対値を求め，フローティング・アキュームレータに格納します．

▶アドレス　２６７ＥＨ

名称　単精度実数型数値の符号反転

レジスタ　A，F，H，L

内容　フローティング・アキュームレータ内の単精度実数型数値の符号を反転します．
実際にはＦ０ＡＡＨ番地の最上位ビットを反転する事によって，符号の反転を行なっています．

▶アドレス　２６８６Ｈ

名称　ＳＧＮエントリ

レジスタ　A，F，H，L

内容　フローティング・アキュームレータ内の数値データの符号が正の場合１を，０の場合０を，負の場合には－１をフローティング・アキュームレータに整数形式で格納し，ＥＦ４５Ｈ番地にも２が入ります．
ＥＦ４５Ｈ番地に３が入っている場合などは，タイプ・ミスマッチのエラーを出力してＮ－ＢＡＳＩＣに制御を移します．

▶アドレス　２６８９Ｈ

名称　８ビット整数収納

レジスタ　A，F，H，L

内容　Ａレジスタの符号付８ビットの整数（－１２８～１２７）を，符号付１６ビットの整数に変換して，フローティング・アキュームレータに格納し，ＥＦ４５Ｈ番地にも，整数型を示す２を入れます．

▶アドレス　２６ＡＦＨ

名称　単精度実数型数値の移動

レジスタ　B，C，D，E，H，L

内容　ＨＬレジスタ対の指定するアドレスから４バイトに渡る単精度実数型の数値データを，ＥＤＣＢレジスタとフローティング・アキュームレータの両方に格納します．

▶アドレス　２６Ｂ２Ｈ

名称　単精度実数型数値の移動

レジスタ　D，E

内容　ＥＤＣＢレジスタ内の単精度実数型数値を，フローティング・アキュームレータに移動し，ＥＦ４５Ｈ番地にも４が入ります．

▶アドレス　２６ＢＤＨ

名称　単精度実数型数値の移動

レジスタ　B，C，D，E，H，L

内容　フローティング・アキュームレータ内の単精度実数型数値を，ＥＤＣＢレジスタに移動します．

▶アドレス　２６Ｃ０Ｈ

名称　単精度実数型数値の移動

レジスタ　B，C，D，E，H，L

内容　ＨＬレジスタ対で指定するアドレスから４バイトに渡る単精度実数型の数値データを，ＥＤＣＢレジスタに移動します．

▶アドレス　２６Ｃ９Ｈ

名称　単精度実数型数値の移動

レジスタ　A，B，C，D，E，H，L

内容　フローティング・アキュームレータ内の単精度実数型数値を，ＨＬレジスタ

対で指定するアドレスから4バイトのメモリ上に移動します.

▶アドレス　26D5H

名称　ブロック転送

レジスタ　A, F, B, D, E, H, L

内容　DEレジスタ対にスタート・アドレスを，HLレジスタ対に転送先のスタート・アドレスを，Bレジスタに転送するデータのバイト数を入れてコールします.

例　0000H番地から256バイトのデータを，C000H番地以降に転送する.

```
LD      DE, 0000H
LD      HL, C000H
LD      B, 0
CALL    26D5H
RET
```

▶アドレス　270CH

名称　単精度実数型数値の比較

レジスタ　A, F, H, L

内容　EDCBレジスタ内の単精度実数型数値Xと，フローティング・アキュームレータ内の単精度実数型数値Yを比較します.
比較結果はAレジスタ（Acc.），キャリー・フラグ（CY），ゼロ・フラグ（Z）にセットされます.

比較結果	Acc	CY	Z
X＞Y	ＦＦＨ	1	0
X＝Y	0	0	1
X＜Y	1	0	0

▶アドレス　２７３９Ｈ

名称　整数型数値の比較

レジスタ　A，F

内容　ＬＨレジスタ内の整数型数値とＥＤレジスタ内の整数型数値を比較します.
比較結果は，Ａレジスタ（Ａｃｃ．），キャリー・フラグ（ＣＹ），ゼロ・フラグ（Ｚ）にセットされます.

比較結果	Acc	ＣＹ	Z
ＬＨ＞ＥＤ	ＦＦＨ	1	0
ＬＨ＝ＥＤ	0	0	1
ＬＨ＜ＥＤ	1	0	0

▶アドレス　２７７８Ｈ

名称　倍精度実数型数値の比較

レジスタ　A，F，D，E

内容　フローティング・アキュームレータ内の倍精度実数型数値Ｘと，サブ・フローティング・アキュームレータであるＦ０ＡＥＨ～Ｆ０Ｂ５Ｈ番地に渡る倍精度実数型数値Ｙを比較します.
比較結果はＡレジスタ（Ａｃｃ．），キャリー・フラグ（ＣＹ），ゼロ・フラグ（Ｚ）にセットされます.

比較結果	Acc.	CY	Z
X＞Y	FFH	1	0
X＝Y	0	0	1
X＜Y	1	0	0

▶アドレス　277FH

名称　CINTエントリ

レジスタ　A, F, B, C, D, E, H, L

内容　フローティング・アキュームレータ内の数値データが整数型でなければ整数型に変換し，同時にEF45H番地にも整数型を示す2が入ります．
EF45H番地に3が入っている場合などは，タイプ・ミスマッチのエラーを出力してN−BASICに制御を移します．

▶アドレス　279CH

名称　整数型数値の格納

レジスタ　A

内容　HLレジスタ対のデータを整数型定数として，フローティング・アキュムレータ内の，F0A8H〜F0A9H番地に格納し，同時にEF45H番地にも整数型を示す2が入ります．

▶アドレス　27B3H

名称　CSNGエントリ

レジスタ　A, F, B, C, D, E, H, L

内容　フローティング・アキュームレータ内の数値データが単精度型でなければ単精度型に変換し，同時にＥＦ４５Ｈ番地にも単精度型を示す4が入ります.
ＥＦ４５Ｈ番地に3が入っている場合などは，タイプ・ミスマッチのエラーを出力してＮ－ＢＡＳＩＣに制御を移します.

▶アドレス　２７ＤＦＨ

名称　ＣＤＢＬエントリ

レジスタ　A, F, B, C, D, E, H, L

内容　フローティング・アキュームレータ内の数値データが倍精度実数型でなければ倍精度実数型に変換し，同時にＥＦ４５Ｈ番地にも倍精度実数型を示す8が入ります.
ＥＦ４５Ｈ番地に3が入っている場合などは，タイプ・ミスマッチのエラーを出力してＮ－ＢＡＳＩＣに制御を移します.

▶アドレス　２８２ＣＨ

名称　ＦＩＸエントリ

レジスタ　A, F, B, C, D, E, H, L

内容　フローティング・アキュームレータ内の数値データの整数部分をフローティング・アキュームレータに求め，ＥＦ４５Ｈ番地にも型を示す数字が入ります.
ＥＦ４５Ｈ番地に3が入っている場合などは，タイプ・ミスマッチのエラー

を出力してN－BASICに制御を移します.

▶アドレス　283FH

名称　INTエントリ

レジスタ　A, F, B, C, D, E, H, L

内容　フローティング・アキュームレータ内の数値データを越えない最大の整数を求め，フローティング・アキュームレータに格納し，EF45H番地にも型を示す数字が入ります.
EF45H番地に3が入っている場合などは，タイプ・ミスマッチのエラーを出力してN－BASICに制御を移します.

▶アドレス　28D2H

名称　整数型数値の減算

レジスタ　A, F, B, C, D, E, H, L

内容　EDレジスタ内の整数型数値からLHレジスタ内の整数型数値を減じた結果をフローティング・アキュムレータに格納します.

▶アドレス　28DDH

名称　整数型数値の加算

レジスタ　A, F, B, C, D, E, H, L

内容　EDレジスタ内の整数型数値にLHレジスタ内の整数型数値を加えた結果を

フローティング・アキュームレータに格納します.
演算の結果が単精度実数型になれば，単精度実数型として格納し，ＥＦ４５Ｈ番地にも単精度実数型を示す４が入ります.

▶アドレス　２８ＦＤＨ

名称　整数型数値の乗算

レジスタ　A, F, B, C, D, E, H, L

内容　ＥＤレジスタ内の整数型数値にＬＨレジスタ内の整数型数値を乗じた結果をフローティング・アキュームレータに格納します.
演算の結果が単精度実数型になれば，単精度実数型として格納し，ＦＥ４５Ｈ番地にも単精度実数型を示す４が入ります.

▶アドレス　２９Ｃ３Ｈ

名称　倍精度実数型数値の減算

レジスタ　A, F, B, C, D, E, H, L

内容　フローティング・アキュームレータ内の倍精度実数型数値から，サブ・フローティング・アキュームレータである，Ｆ０ＡＥＨ～Ｆ０Ｂ５Ｈ番地に渡る倍精度実数型数値を減じて差をフローティング・アキュームレータに格納します.

▶アドレス　２９ＣＡＨ

名称　倍精度実数型数値の加算

レジスタ　A, F, B, C, D, E, H, L

内容　フローティング・アキュームレータ内の倍精度実数型数値に，サブ・フローティング・アキュームレータである，F0AEH～F0B5H番地に渡る倍精度実数型数値を加えて，和をフローティング・アキュームレータに格納します.

▶アドレス　2A2BH

名称　倍精度実数型数値の正規化と符号付け

レジスタ　A, F, B, C, H, L

内容　フローティング・アキュームレータの倍精度実数型の数値を正規化し，F0ACH番地の最上位ビットが1であれば正符号を，0であれば負符号を付けます.

▶アドレス　2AF4H

名称　倍精度実数型数値の乗算

レジスタ　A, F, B, C, D, E, H, L

内容　フローティング・アキュームレータ内の倍精度実数型数値に，サブ・フローティング・アキュームレータである，F0AEH～F0B5H番地に渡る倍精度実数型数値を乗じて積をフローティング・アキュームレータに格納します.

▶アドレス　2B37H

名称　倍精度実数型数値の除算

レジスタ　A，F，B，C，D，E，H，L

内容　フローティング・アキュームレータ内の倍精度実数型数値を，サブ・フローティング・アキュームレータである，F０AEH～F０B５H番地に渡る倍精度実数型数値で割り，商をフローティング・アキュームレータに格納します．

▶アドレス　2BBEH

名称　キャラクタ・コード→数値データ変換

レジスタ　A，F，B，C，D，E

内容　HLレジスタ対で示すアドレスから０が入っているアドレスまでにキャラクタ・コードで格納されている数値データを，整数型，単精度実数型，倍精度実数型のいずれかに変換して，フローティング・アキュームレータに格納し，EF４５H番地にも型を示す数字を入れます．

▶アドレス　2D0BH

名称　 in 行番号表示

レジスタ　A，F，B，C，D，E，H，L

内容　 in をCRT画面に表示した後，HLレジスタ対のデータを符号なしの１０進数として表示します．
N－BASICでは，ブレーク時，エラー時などの発生行番号を表示するために，このサブルーチンを呼んでいます．

▶アドレス　2D13H

名称　16ビット10進出力

レジスタ　A, F, B, C, D, E, H, L

内容　HLレジスタ対の2バイト・データを符号なし5桁の10進数として，CRT画面上のカーソル位置に表示します.

例　フローティング・アキュームレータの整数型数値を符号なし10進数として表示する.

```
LD      HL, (F0A8H)
CALL    2D13H
RET
```

▶アドレス　2D22H

名称　数値データ→キャラクタ・コード変換

レジスタ　A, F, B, C, D, E, H, L

内容　EF45H番地の型を示す数字により，フローティング・アキュームレータ内の数値をキャラクタ・コードの文字列に変換してF0B7H番地からのバッファに収納し，最後に0を入れます.
HLレジスタ対には，文字列の先頭アドレスが入ります.

例　フローティング・アキュームレータの数値データをCRT画面に表示します.

```
CALL    2D22H
CALL    52EDH
RET
```

▶アドレス　309FH

名称　16進→10進文字列変換

レジスタ　A, F, B, C, H, L

内容　フローティング・アキュームレータの整数型数値データを，符号なし5桁の10進文字列に変換してHLレジスタ対で指定したアドレスからキャラクタ・コードで格納し最後に0を入れます.
ただし，ゼロ・サプレスは行なわず，Cレジスタが1の時にカンマ「，」を付け，Bレジスタが1の時には先頭に小数点「．」をつけます.

例　フローティング・アキュームレータの整数型数値データを5桁の10キャラクタ・コードに変換します.

```
CALL  309FH
RET
```

▶アドレス　3148H～3151H（DATA）

名称　16進→10進変換データ

レジスタ　――――

内容　16進数を10進数に変換するためのデータで，309FH番地からのサブルーチンで使用しています.
データは，2バイトの定数5組から成り，それぞれ，以下の様に対応しています.

```
2710H ……→ 10000
03E8H ……→  1000
0064H ……→   100
000AH ……→    10
```

０００１H ………→　　　　　１

▶アドレス　３１A１H

名称　SQRエントリ

レジスタ　A, F, B, C, D, E, H, L

内容　フローティング・アキュームレータ内の単精度実数型数値の平方根を求めて，フローティング・アキュームレータに格納します.
入力パラメータが負数の場合には，イリーガル・ファンクション・コールのエラーを出力して，N－BASICに制御を移します.

▶アドレス　３１F３H

名称　EXPエントリ

レジスタ　A, F, B, C, D, E, H, L

内容　自然対数の底eを，フローティング・アキュームレータ内の単精度実数乗し，結果をフローティング・アキュームレータに格納します.
入力パラメータが，８７．３３６６未満でなければオーバー・フローを起こすので注意が必要です.

▶アドレス　３２８３H

名称　RNDエントリ

レジスタ　A, F, B, C, D, E, H, L

内容　フローティング・アキュームレータの単精度実数型数値が正で有れば，単精度実数型の乱数を発生してフローティング•アキュームレータに格納します．

▶アドレス　３２Ｆ６Ｈ

名称　ＣＯＳエントリ

レジスタ　A，F，B，C，D，E，H，L

内容　フローティング・アキュームレータの単精度実数型数値の単位をラジアンとした場合のコサインの値を求めフローティング・アキュームレータに格納します．

▶アドレス　３３ＢＤＨ～３４６６Ｈ（ＤＡＴＡ）

名称　中間言語→処理ルーチン・テーブル

レジスタ　────

内容　中間言語に対応した処理ルーチンのエントリ・アドレスを求めるためのテーブルで，テーブルのトップ•アドレスは，ワーク•エリア中のＥＡ５２Ｈ～ＥＡ５３Ｈ番地に格納されています．

▶アドレス　３４６７Ｈ～３４Ｃ０Ｈ（ＤＡＴＡ）

名称　関数の中間言語→処理ルーチン・テーブル

レジスタ　────

内容　関数の中間言語に対応した処理ルーチンのエントリ・アドレスを求めるためのテーブルです．

▶**アドレス**　３４Ｃ１Ｈ～３７Ａ７Ｈ（ＤＡＴＡ）

名称　キーワード→中間言語変換テーブル

レジスタ　――――

内容　Ｎ―ＢＡＳＩＣのコマンド，ステートメント，及び，関数のキーワードを中間言語に変換するためのテーブルです．

▶**アドレス**　３７Ａ８Ｈ～３７ＢＣＨ（ＤＡＴＡ）

名称　特殊記号→中間言語変換テーブル

レジスタ　――――

内容　特殊記号を中間言語に変換するためのテーブルです．

▶**アドレス**　３７Ｆ２Ｈ～３Ａ０４Ｈ（ＤＡＴＡ）

名称　エラー・メッセージ・テーブル

レジスタ　――――

内容　エラー・コードの１から２９に対応するエラー・メッセージが格納されています．各エラー・メッセージの区切りは０です．

▶アドレス　３Ａ０５Ｈ～３Ｂ５ＢＨ（ＤＡＴＡ）

名称　ワーク・エリア初期データ

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのコールド・スタート時にワーク・エリアのＥＡ００Ｈ～ＥＢ５６Ｈ番地に転送されるデータで，その内３Ａ８１Ｈ～３Ｂ２０Ｈ番地には，ファンクション・キー・バッファの初期定義内容が入っています．

▶アドレス　３ＢＦ１Ｈ

名称　Ｏｖｅｒｆｌｏｗエラー出力

レジスタ　――――

内容　Ｏｖｅｒｆｌｏｗエラーを出力してＮ－ＢＡＳＩＣに制御を移します．

▶アドレス　３ＢＦ９Ｈ

名称　エラー出力

レジスタ　――――

内容　Ｅレジスタのエラー・コードに対応するエラー・メッセージを出力して，Ｎ－ＢＡＳＩＣに制御を移します．
Ｅレジスタのコードが指定外の場合には，

Ｕｎｐｒｉｎｔａｂｌｅ　ｅｒｒｏｒ

を出力し，０の場合には無限ループに入り込みます．

例　Ｐｏｓｉｔｉｏｎ　ｎｏｔ　ｏｎ　ｓｃｒｅｅｎエラーを出力する．

```
LD    E, 18H
JP    3BF9H
```

▶アドレス　３Ｄ７６Ｈ

名称　Ｎ－ＢＡＳＩＣテキスト・サーチ

レジスタ　Ａ，Ｆ，Ｄ，Ｅ，Ｈ，Ｌ

内容　ＥＢ５４Ｈ～ＥＢ５５Ｈ番地に格納されている，プログラム・エリアの先頭アドレスから，ＨＬレジスタ対をポインタとして，プログラム・エリアを上位に向かってサーチして行き次の行の先頭アドレスを見つけたら現在行のリンク・ポインタに代入し，以上の作業を，次の行の先頭の２バイトが０になるまで続けます．ＨＬレジスタ対には，サーチを終えたアドレスが代入されてくるため，プログラムのエンド・アドレスを見付けるためにサブルーチンを使用する事ができます．
ＤＥレジスタ対には，ＨＬレジスタ対の値から１を減じたものが入ります．

▶アドレス　３Ｄ７９Ｈ

名称　Ｎ－ＢＡＳＩＣテキスト・サーチ

レジスタ　Ａ，Ｆ，Ｄ，Ｅ，Ｈ，Ｌ

内容　ＨＬレジスタ対で示すアドレスから，プログラム・エリアを上位に向かってサーチして行き次の行の先頭アドレスを見つけたら現在行のリンク・ポインタに代入し，以上の作業を，次の行の先頭の２バイトが０になるまで続けます．ＨＬレジスタ対には，サーチを終えたアドレスが代入されてくるた

め，プログラムのエンド・アドレスを見付けるためにサブルーチンを使用する事ができます．

ＤＥレジスタ対には，ＨＬレジスタ対の値から１を減じたものが入ります．

▶アドレス　３ＤＥ０Ｈ

名称　ＮＥＷエントリ

レジスタ　――――

内容　Ａレジスタに０を入れてジャンプする事によって，Ｎ－ＢＡＳＩＣのテキスト・プログラムを消去する事ができます．

Ｎ－ＢＡＳＩＣのテキストを消去する場合には実際のプログラム・エリアはそのまま残っており，先頭のリンク・ポインタとＥＦＡ０Ｈ～ＥＦＡ５Ｈ番地の内容が変更されるだけですから，これらの値をもとにもどす事によって元のプログラムを復活する事ができます．

例　Ｎ－ＢＡＳＩＣテキストを消去する．

```
XOR  A
JP   3DE0H
```

▶アドレス　３Ｅ５ＣＨ

名称　インプット

レジスタ　Ａ，Ｆ，Ｂ，Ｃ，Ｄ，Ｅ，Ｈ，Ｌ

内容　入力を促すプロンプト「？」をＣＲＴ画面上のカーソル位置に表示して，ＣＲＴへのエコー・バックを行ないながらキーボードからリターン・キーが押されるまで入力を行ないます．

入力データは，ＥＣ９６Ｈ～ＥＤ９５Ｈ番地にあるバッファにキャラクタ・コードで収納し，データの最後には0が，ＨＬレジスタ対にはＥＣ９５Ｈが入ります。

例　インプットを行なった後，入力データ全体を再度エコー・バックする.

```
CALL  3E5CH
CALL  52ECH
RET
```

▶アドレス　４０９５Ｈ

名称　ペア・レジスタ比較

レジスタ　A，F

内容　ＨＬレジスタ対とＤＥレジスタ対のデータが等しければ，ゼロ・フラグ（Ｚ）を１にセットし，等しく無ければリセットしてもどります.

▶アドレス　４０Ａ６Ｈ

名称　周辺機器への１バイト出力

レジスタ　全て保存

内容　ＥＢ４９Ｈ番地の値によって，各周辺機器へＡレジスタのデータ１バイトを出力します．通常はＣＲＴ画面への１文字表示等に用いられます.
又，ＣＭＴへの出力では，事前にＣＭＴをイニシアライズする事が必要です.

ＥＢ４９Ｈ番地	周辺機器
００Ｈ	ＣＲＴ
０１Ｈ～７ＦＨ	プリンタ
８０Ｈ～ＦＦＨ	ＣＭＴ

▶アドレス　４１５９Ｈ

名称　ＦＯＲエントリ

レジスタ　————

内容　Ｎ－ＢＡＳＩＣのＦＯＲエントリ・アドレスです.

▶アドレス　４１ＦＡＨ

名称　コントロールＢ

レジスタ　————

内容　ＦＦ３４Ｈ～ＦＦ３５Ｈ番地に退避してあった，Ｎ－ＢＡＳＩＣのテキスト・ポインタをＨＬレジスタ対にもどしてジャンプする事によって，Ｎ－ＢＡＳＩＣにリターンする事ができます.
Ｎ－ＢＡＳＩＣでは，ＨＬレジスタ対をテキスト・ポインタとして用いています.

例　Ｎ－ＢＡＳＩＣへリターンする.

```
LD      HL,（FF34H）
JP      41FAH
```

▶アドレス　３２ＦＣＨ

名称　ＳＩＮエントリ

レジスタ　A，F，B，C，D，E，L

内容　フローティング・アキュームレータの単精度実数をラジアンとして，サインを計算し，フローティング・アキュームレータに格納します.

▶アドレス　３３５ＤＨ

名称　ＴＡＮエントリ

レジスタ　A，F，B，C，D，E，H，L

内容　フローティング・アキュームレータの単精度実数型数値の単位をラジアンとした場合のタンジェントを求めて，フローティング・アキュームレータに格納します.
演算結果がオーバー・フローした場合にはエラー・メッセージを出力してＮ－ＢＡＳＩＣに制御を移します.

▶アドレス　３３７２Ｈ

名称　ＡＴＮエントリ

レジスタ　A，F，B，C，D，E，H，L

内容　フローティング・アキュームレータの単精度実数型数値の単位をラジアンとした場合のアーク・タンジェントの値を求めて，フローティング・アキュームレータに格納します.
演算結果は，$-\pi/2$から$\pi/2$の範囲になります.

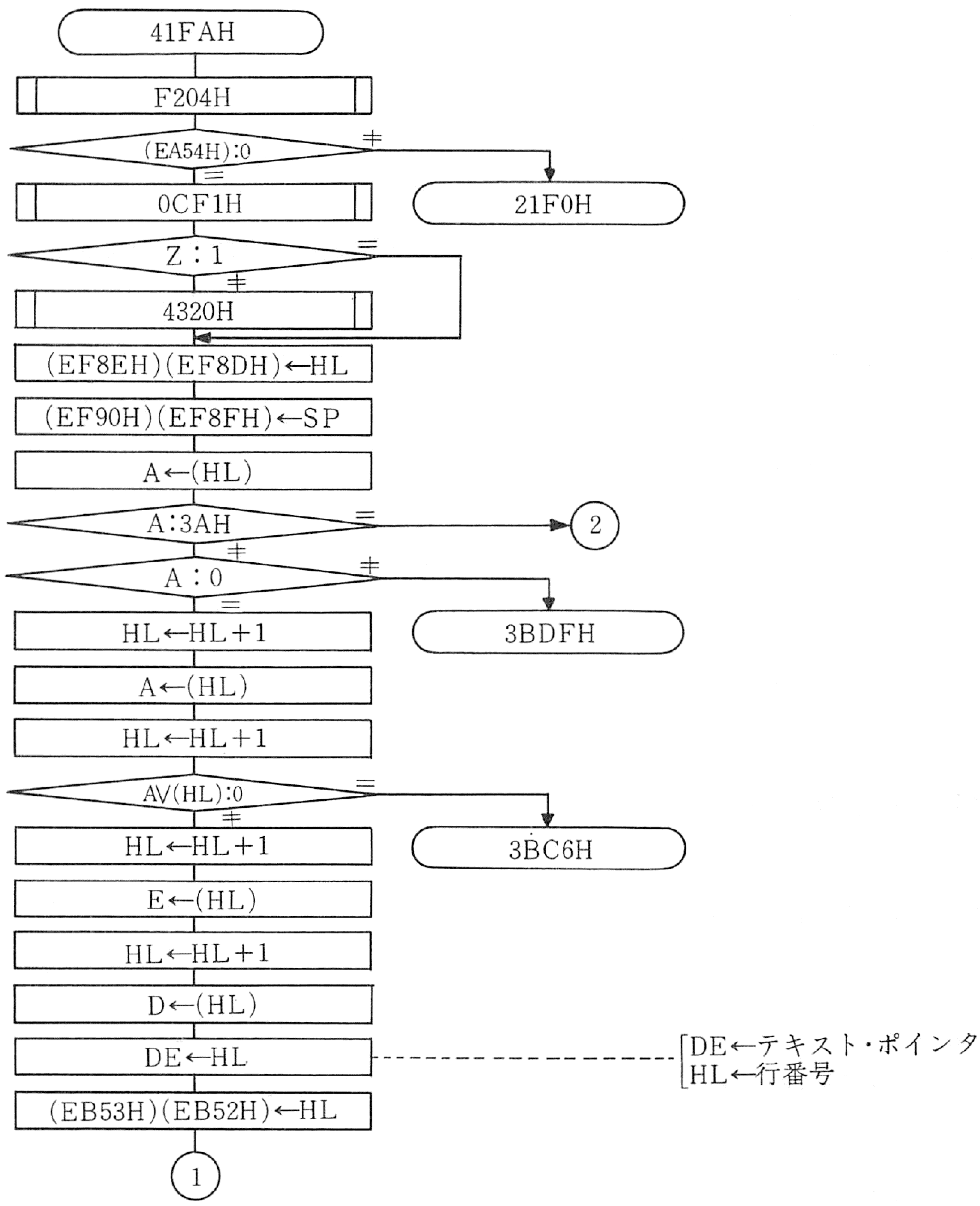
▶アドレス　４１ＦＡＨ

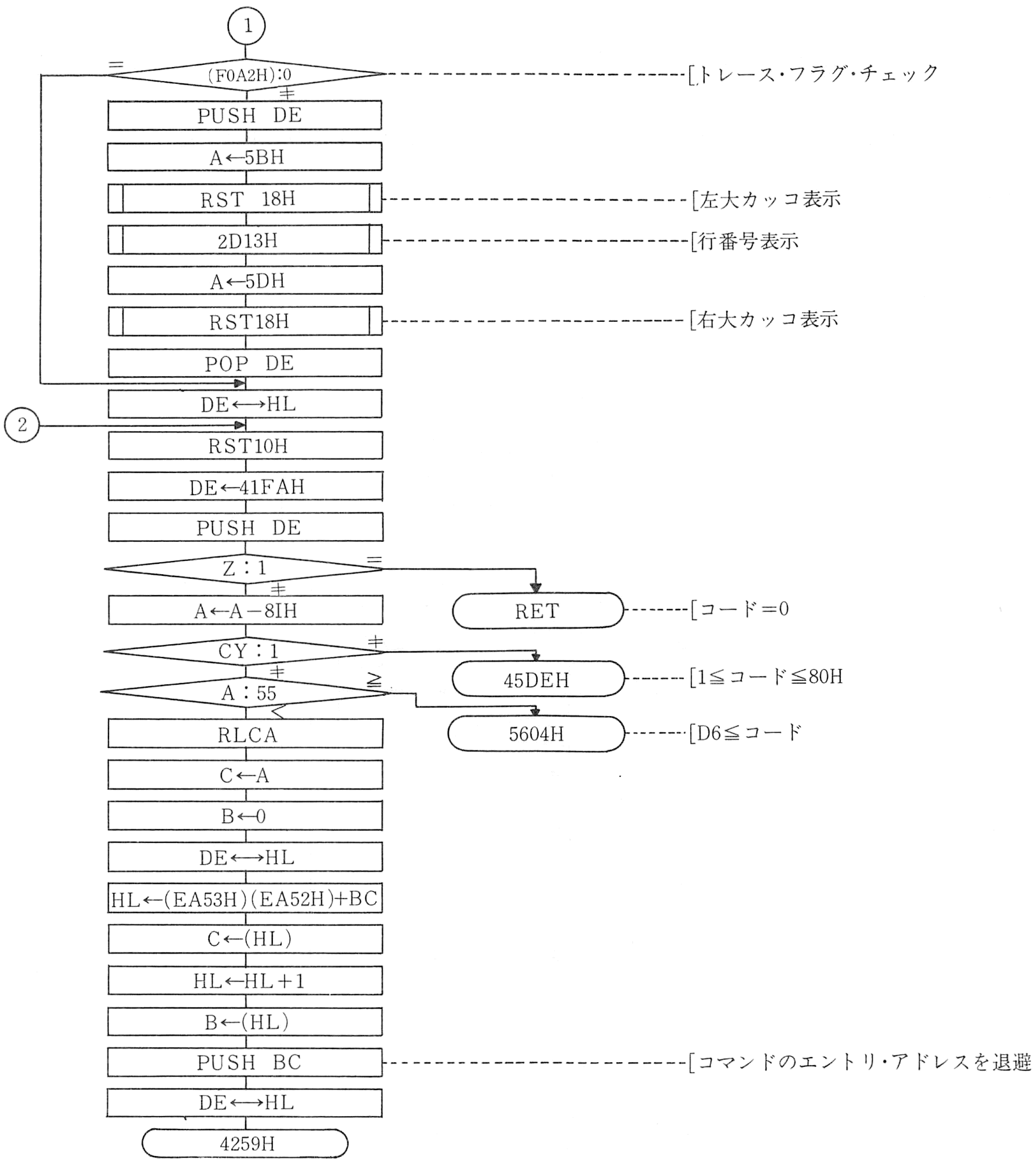
名称　実行

レジスタ　————

内容　中間言語に直されたＮ－ＢＡＳＩＣのテキストをステートメント単位で実行します．

フロー・チャート





▶アドレス　4 2 5 9 H

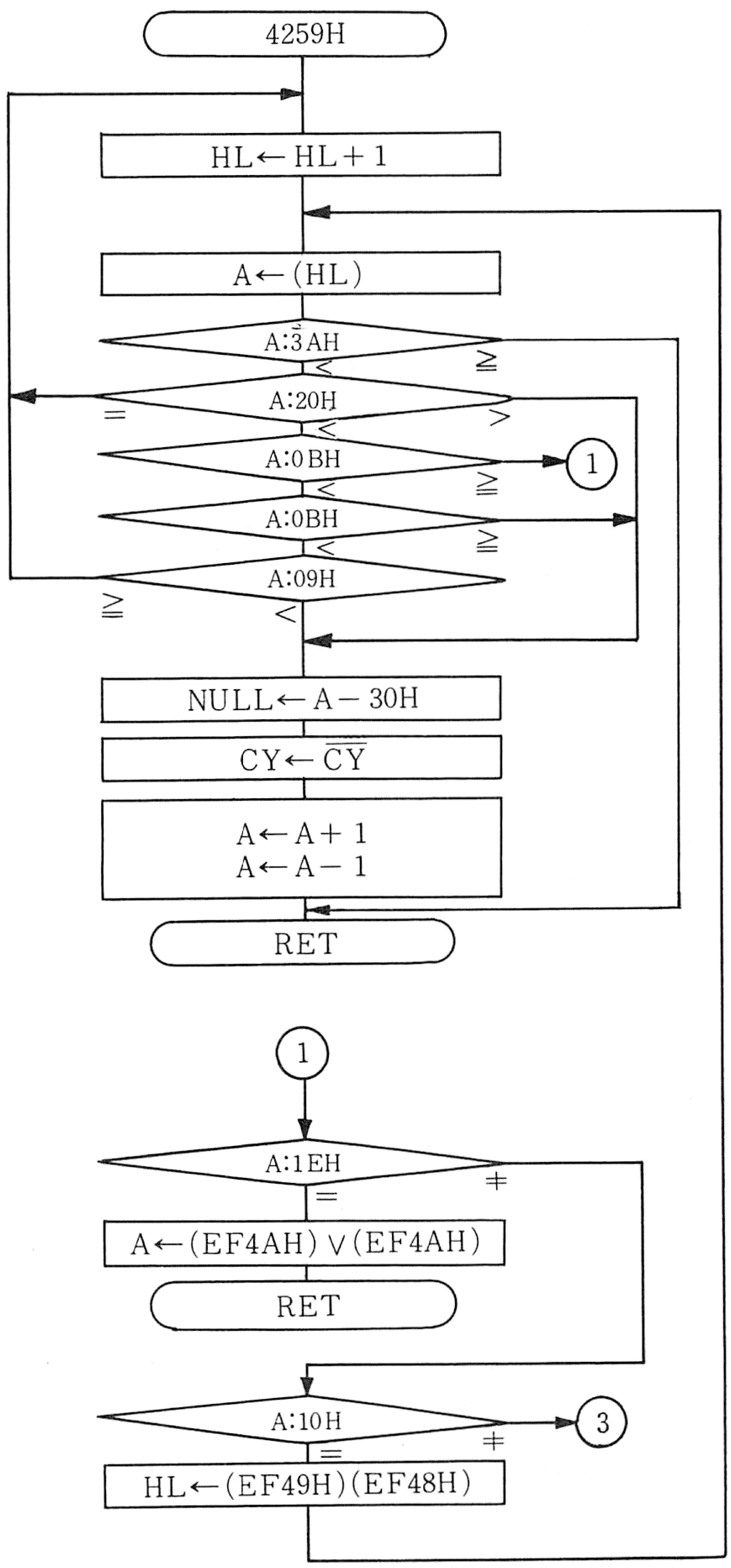
名称　N－B A S I Cテキスト解釈

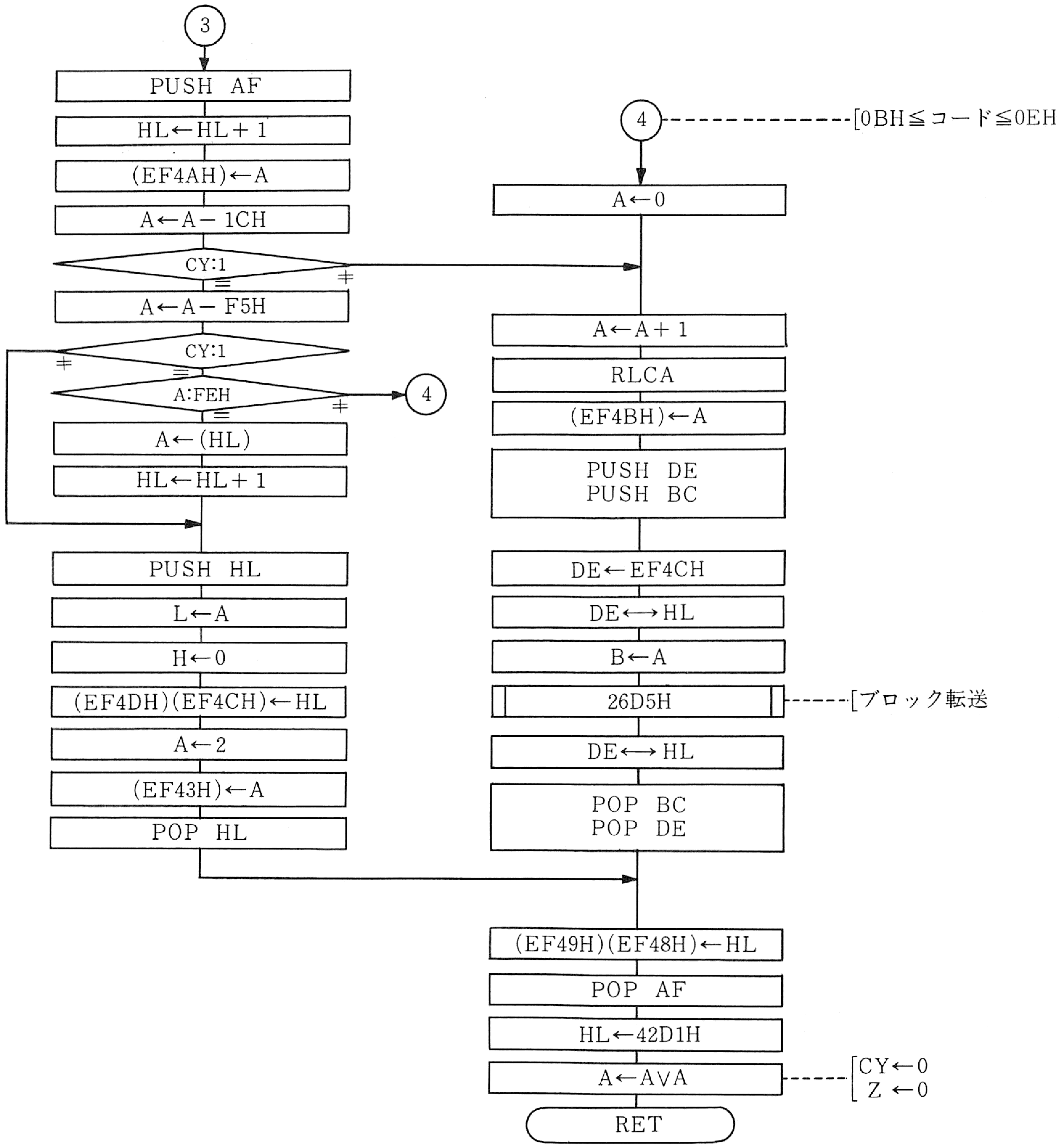
レジスタ　A, F, B, C, D, E, H, L

内容　RST　10Hによってコールされ，N－BASICのテキストを一文字ずつ解釈して行くためのサブルーチンです．

詳しくは，フロー・チャートを参照してください．

フロー・チャート





▶アドレス　4302H

名称　RESTOREエントリ

レジスタ　――――

内容　N－BASICのRESTOREエントリ・アドレスです.

▶アドレス　432AH

名称　STOPエントリ

レジスタ　――――

内容　N－BASICのSTOPエントリ・アドレスです.

▶アドレス　432FH

名称　ENDエントリ

レジスタ　――――

内容　N－BASICのENDエントリ・アドレスです.

▶アドレス　4383H

名称　CONTエントリ

レジスタ　――――

内容　N－BASICのCONTエントリ・アドレスです.

▶アドレス　4396H

名称　ＴＲＯＮエントリ

レジスタ　————

内容　Ｎ－ＢＡＳＩＣのＴＲＯＮエントリ・アドレスです.

▶アドレス　４３９７Ｈ

名称　ＴＲＯＦＦエントリ

レジスタ　————

内容　Ｎ－ＢＡＳＩＣのＴＲＯＦＦエントリ・アドレスです.

▶アドレス　４３９ＣＨ

名称　ＳＷＡＰエントリ

レジスタ　————

内容　Ｎ－ＢＡＳＩＣのＳＷＡＰエントリ・アドレスです.

▶アドレス　４３ＤＦＨ

名称　ＥＲＡＳＥエントリ

レジスタ　————

内容　Ｎ－ＢＡＳＩＣのＥＲＡＳＥエントリ・アドレスです.

▶アドレス　４４５ＢＨ

名称　ＤＥＦＳＴＲエントリ

レジスタ　――――

内容　N―BASICのDEFSTRエントリ・アドレスです.

▶アドレス　４４５ＥＨ

名称　ＤＥＦＩＮＴエントリ

レジスタ　――――

内容　N―BASICのDEFINTエントリ・アドレスです.

▶アドレス　４４６１Ｈ

名称　ＤＥＦＳＮＧエントリ

レジスタ　――――

内容　N―BASICのDEFSNGエントリ・アドレスです.

▶アドレス　４４６４Ｈ

名称　ＤＥＦＤＢＬエントリ

レジスタ　――――

内容　N—BASICのDEFDBLエントリ・アドレスです.

▶アドレス　**44A5H**

名称　**Illegal function callエラー出力**

レジスタ　————

内容　Illegal function callエラーを出力してN—BASICに制御を移します.

例　Illegal function callエラーを出力する.

```
JP      44A5H
```

▶アドレス　**44C7H**

名称　**10進文字列→16進数値変換**

レジスタ　A, F, D, E

内容　HLレジスタ対に10進文字列の先頭アドレス－1を入れてコールすると，16進数値に変換したものがDEレジスタ対に格納されてもどります.
ただし，10進文字列は0から65529までの整数でなければならず，N—BASICでは行番号の管理等にこのサブルーチンを用いています.

例　30000を4桁の16進数に変換して，DEレジスタ対に格納します.

```
        LD      HL, LBL—1
        CALL    44C7H
        RET
LBL:DB          '30000 '
```

▶アドレス　４４Ｅ８Ｈ

名称　ＣＬＥＡＲエントリ

レジスタ　――――

内容　Ｎ―ＢＡＳＩＣのＣＬＥＡＲエントリ・アドレスです.

▶アドレス　４５３ＤＨ

名称　ＲＵＮエントリ

レジスタ　――――

内容　Ｎ―ＢＡＳＩＣのＲＵＮエントリ・アドレスです.

▶アドレス　４５５５Ｈ

名称　ＧＯＳＵＢエントリ

レジスタ　――――

内容　Ｎ―ＢＡＳＩのＧＯＳＵＢエントリ・アドレスです.

▶アドレス　４５６ＤＨ

名称　ＧＯＴＯおよびＧＯ　ＴＯエントリ

レジスタ　――――

内容　N－BASICのGOTOおよびGO　TOエントリ・アドレスです.

▶アドレス　45A3H

名称　RETURNエントリ

レジスタ　――――

内容　N－BASICのRETURNエントリ・アドレスです.

▶アドレス　45BEH

名称　DATAエントリ

レジスタ　――――

内容　N－BASICのDATAエントリ・アドレスです.

▶アドレス　45C0H

名称　ELSEおよびREMエントリ

レジスタ　――――

内容　N－BASICのELSEおよびREMエントリ・アドレスです.

▶アドレス　45DEH

名称　ＬＥＴエントリ

レジスタ　────

内容　Ｎ―ＢＡＳＩＣのＬＥＴエントリ・アドレスです．

▶アドレス　４６４２Ｈ

名称　ＯＮエントリ

レジスタ　────

内容　Ｎ―ＢＡＳＩＣのＯＮエントリ・アドレスです．

▶アドレス　４６８ＣＨ

名称　ＲＥＳＵＭＥエントリ

レジスタ　────

内容　Ｎ―ＢＡＳＩＣのＲＥＳＵＭＥエントリ・アドレスです．

▶アドレス　４６Ｃ４Ｈ

名称　ＥＲＲＯＲエントリ

レジスタ　────

内容　Ｎ―ＢＡＳＩＣのＥＲＲＯＲエントリ・アドレスです．

▶アドレス　４６ＣＦＨ

名称　ＡＵＴＯエントリ

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのＡＵＴＯエントリ・アドレスです.

▶アドレス　４７０２Ｈ

名称　ＩＦエントリ

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのＩＦエントリ・アドレスです.

▶アドレス　４７３ＡＨ

名称　ＬＰＲＩＮＴエントリ

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのＬＰＲＩＮＴエントリ・アドレスです.

▶アドレス　４７４２Ｈ

名称　ＰＲＩＮＴエントリ

レジスタ　――――

内容　N－BASICのPRINTエントリ・アドレスです.

▶アドレス　4877H

名称　LINEエントリ

レジスタ　――――

内容　N－BASICのLINEエントリ・アドレスです.

▶アドレス　48DAH

名称　INPUTエントリ

レジスタ　――――

内容　N－BASICのINPUTエントリ・アドレスです.

▶アドレス　4939H

名称　READエントリ

レジスタ　――――

内容　N－BASICのREADエントリ・アドレスです.

▶アドレス　4A08H

名称　ＮＥＸＴエントリ

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのＮＥＸＴエントリ・アドレスです．

▶アドレス　４ＢＥＡＨ

名称　整数型数値の除算

レジスタ　Ａ，Ｆ，Ｂ，Ｃ，Ｄ，Ｅ，Ｈ，Ｌ

内容　ＥＤレジスタ内の整数型数値をＬＨレジスタ内の整数型数値で割った結果をフローティング・アキュームレータに格納します．
演算の結果が単精度実数型になれば，単精度実数型として格納し，ＥＦ４５Ｈ番地にも単精度実数型を示す４が入ります．

▶アドレス　４ＣＣＢＨ

名称　英小文字→英大文字変換

レジスタ　Ａ，Ｆ

内容　ＨＬレジスタ対で指定したアドレスのキャラクタ・コードをＡレジスタに移し，英小文字であれば英大文字に変換します．

▶アドレス　４ＣＣＣＨ

名称　英小文字→英大文字変換

レジスタ　A，F

内容　Aレジスタのキャラクタ・コードが英小文字のものであれば，英大文字に変換します.

▶アドレス　4CE9H

名称　16進文字列→数値データ変換

レジスタ　A，F，B，D，E，H，L

内容　HLレジスタ対で指定する次のアドレスから0が格納されているアドレスまでの16進キャラクタ・コードをバイナリ・コードに変換して，DEレジスタ対と整数型のフローティング・アキュームレータであるF0A8H～F0A9H番地に格納し，EF45H番地にも2が入ります.
ただし，オーバーフローが発生した場合には，エラーを出力してN－BASICに制御を移します.

▶アドレス　4E37H

名称　DIMエントリ

レジスタ　――――

内容　N－BASICのDIMエントリ・アドレスです.

▶アドレス　5051H

名称　FREエントリ

レジスタ　――――

内容　N－BASICのFREエントリ・アドレスです.

▶アドレス　5074H

名称　LPOSエントリ

レジスタ　――――

内容　N－BASICのLPOSエントリ・アドレスです.

▶アドレス　5079H

名称　POSエントリ

レジスタ　――――

内容　N－BASICのPOSエントリ・アドレスです.

▶アドレス　50CCH

名称　DEFエントリ

レジスタ　――――

内容　N－BASICのDEFエントリ・アドレスです.

▶アドレス　５２７０Ｈ

名称　ＯＣＴ＄エントリ

レジスタ　――――――

内容　Ｎ－ＢＡＳＩＣのＯＣＴ＄エントリ・アドレスです．

▶アドレス　５２７５Ｈ

名称　ＨＥＸ＄エントリ

レジスタ　――――――

内容　Ｎ－ＢＡＳＩＣのＨＥＸ＄エントリ・アドレスです．

▶アドレス　５２７ＡＨ

名称　ＳＴＲ＄エントリ

レジスタ　――――――

内容　Ｎ－ＢＡＳＩＣのＳＴＲ＄エントリ・アドレスです．

▶アドレス　５２ＥＣＨ

名称　各周辺機器への文字列出力

レジスタ　Ａ，Ｆ，Ｂ，Ｃ，Ｄ，Ｅ，Ｈ，Ｌ

内容　ＨＬレジスタ対で指定する次のアドレスから，０が入っているアドレスまでのデータをＥＢ４９Ｈ番地の値で指定する周辺機器へ出力します．
通常は，ＣＲＴ画面への表示に使用されますが，ＣＭＴへの出力を行なうためには事前にＣＭＴをイニシアライズする事が必要です．
ただし，１度に２５５文字までしか出力できないので注意が必要です．

ＥＢ４９Ｈ番地	周辺機器
００Ｈ	ＣＲＴ
０１Ｈ～７ＦＨ	プリンタ
８０Ｈ～ＦＦＨ	ＣＭＴ

▶アドレス　５２ＥＤＨ

名称　各周辺機器への文字列出力

レジスタ　A，F，B，C，D，E，H，L

内容　ＨＬレジスタ対で指定するアドレスから，０が入っているアドレスまでのデータをＥＢ４９Ｈ番地の値で指定する周辺機器へ出力します．
通常は，ＣＲＴ画面への表示に使用されますが，ＣＭＴへの出力を行なうためには事前にＣＭＴをイニシアライズする事が必要です．
ただし，一度に２５５文字までしか出力できないので注意が必要です．

ＥＢ４９Ｈ番地	周辺機器
００Ｈ	ＣＲＴ
０１Ｈ～７ＦＨ	プリンタ
８０Ｈ～ＦＦＨ	ＣＭＴ

▶アドレス　５４８ＣＨ

名称　ＬＥＮエントリ

レジスタ　―――――

内容　N－BASICのLENエントリ・アドレスです．

▶アドレス　5498H

名称　ASCエントリ

レジスタ　―――――

内容　N－BASICのASCエントリ・アドレスです．

▶アドレス　54A8H

名称　CHR＄エントリ

レジスタ　―――――

内容　N－BASICのCHR＄エントリ・アドレスです．

▶アドレス　54DFH

名称　SPACE＄エントリ

レジスタ　―――――

内容　N－BASICのSPACE＄エントリ・アドレスです．

▶アドレス　５４Ｆ９Ｈ

名称　ＬＥＦＴ＄エントリ

レジスタ　――――

内容　Ｎ―ＢＡＳＩＣのＬＥＦＴ＄エントリ・アドレスです.

▶アドレス　５５２９Ｈ

名称　ＲＩＧＨＴ＄エントリ

レジスタ　――――

内容　Ｎ―ＢＡＳＩＣのＲＩＧＨＴ＄エントリ・アドレスです.

▶アドレス　５５３２Ｈ

名称　ＭＩＤ＄エントリ

レジスタ　――――

内容　Ｎ―ＢＡＳＩＣのＭＩＤ＄エントリ・アドレスです.

▶アドレス　５５５３Ｈ

名称　ＶＡＬエントリ

レジスタ　――――

内容　N－BASICのVALエントリ・アドレスです.

▶アドレス　５６Ａ１Ｈ

名称　ＩＮＰエントリ

レジスタ　――――

内容　N－BASICのINPエントリ・アドレスです.

▶アドレス　５６ＡＤ

名称　ＯＵＴエントリ

レジスタ　――――

内容　N－BASICのOUTエントリ・アドレスです.

▶アドレス　５６Ｂ３Ｈ

名称　ＷＡＩＴエントリ

レジスタ　――――

内容　N－BASICのWAITエントリ・アドレスです.

▶アドレス　５７０７Ｈ

名称　ＬＬＩＳＴエントリ

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのＬＬＩＳＴエントリ・アドレスです.

▶アドレス　５７０Ｃ

名称　ＬＩＳＴエントリ

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのＬＩＳＴエントリ・アドレスです.

▶アドレス　５８Ｄ９Ｈ

名称　ＤＥＬＥＴＥエントリ

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのＤＥＬＥＴＥエントリ・アドレスです.

▶アドレス　５９１１Ｈ

名称　ＰＥＥＫエントリ

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのＰＥＥＫエントリ・アドレスです.

▶アドレス　５９１８Ｈ

名称　ＰＯＫＥエントリ

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのＰＯＫＥエントリ・アドレスです.

▶アドレス　５ＡＥＤＨ

名称　ＲＥＮＵＭエントリ

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのＲＥＮＵＭエントリ・アドレスです.

▶アドレス　５Ｃ２ＣＨ

名称　マシン語モニタ先頭アドレス

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣ付属のマシン語モニタ先頭アドレスです.
拡張ＲＯＭ領域，７ＦＦＦＨ番地の内容が５５Ｈであれば７ＦＦＣＨ番地にジャンプし，それ以外の場合には，Ｎ－ＢＡＳＩＣのテキスト・ポインタであるＨＬレジスタ対をＦＦ３４Ｈ～ＦＦ３５Ｈ番地に，スタック・ポインタ（ＳＰ）をＦＦ３６Ｈ～ＦＦ３７Ｈ番地に退避して，モニタ・モードに入ります.

▶アドレス　５Ｃ５ＥＨ

名称　マシン語モニタ・エラー

レジスタ　————

内容　エラー・メッセージ・マーク「？」をＣＲＴ画面に表示した後，マシン語モニタのコマンド待ちに移ります．

▶アドレス　５Ｃ６６Ｈ

名称　マシン語モニタ

レジスタ　————

内容　スタック・ポインタ（ＳＰ）を再定義してプロンプト・マーク「＊」をＣＲＴ画面に表示した後，マシン語モニタのコマンド入力を受け付けます．

▶アドレス　５Ｃ６ＦＨ～５Ｃ９２Ｈ（ＤＡＴＡ）

名称　マシン語モニタ・コマンド・テーブル

レジスタ　————

内容　Ｎ－ＢＡＳＩＣ内蔵のマシン語モニタで，キー入力に対応したコマンドのエントリ・アドレスにジャンプするためのテーブルです．

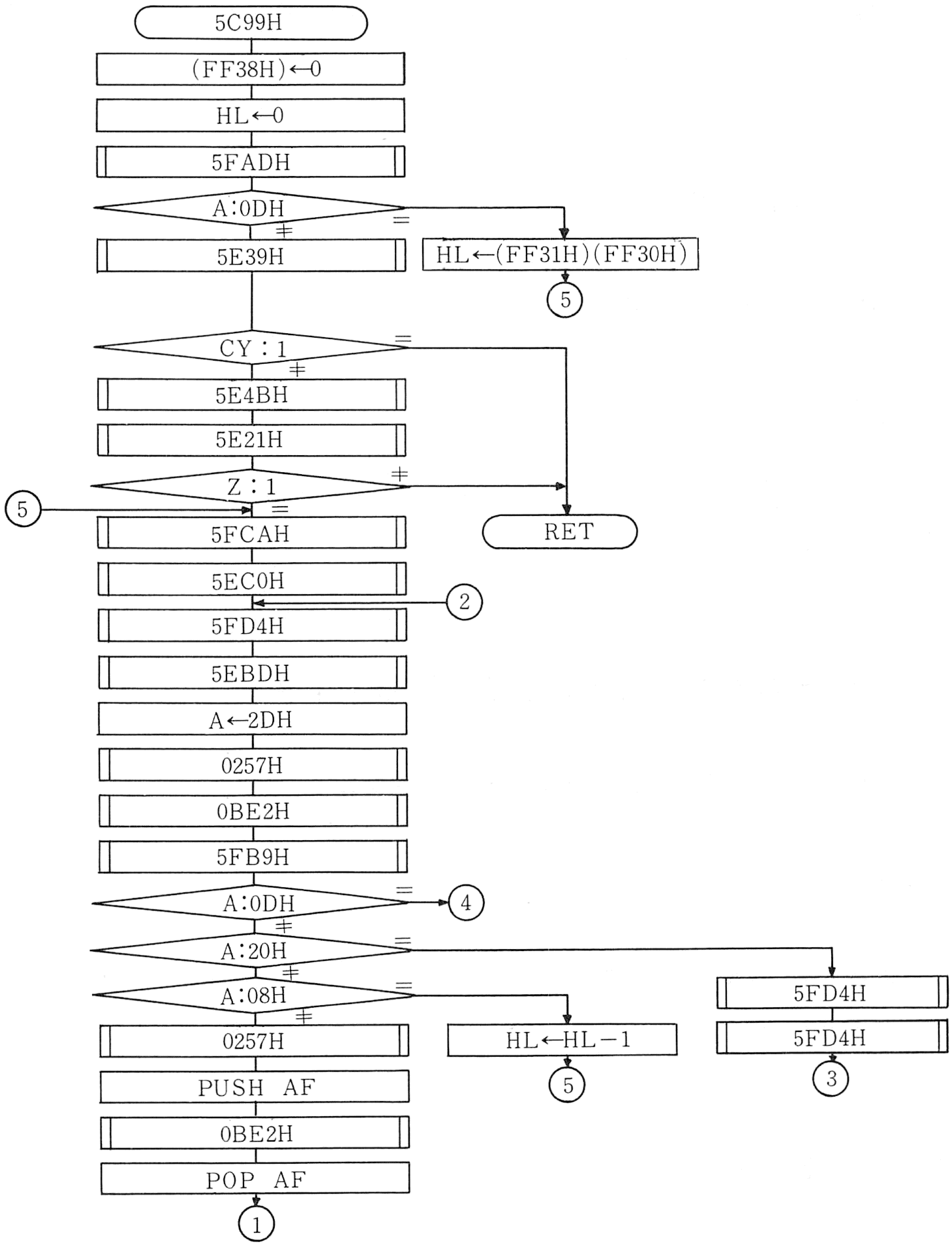
▶アドレス　５Ｃ９９Ｈ

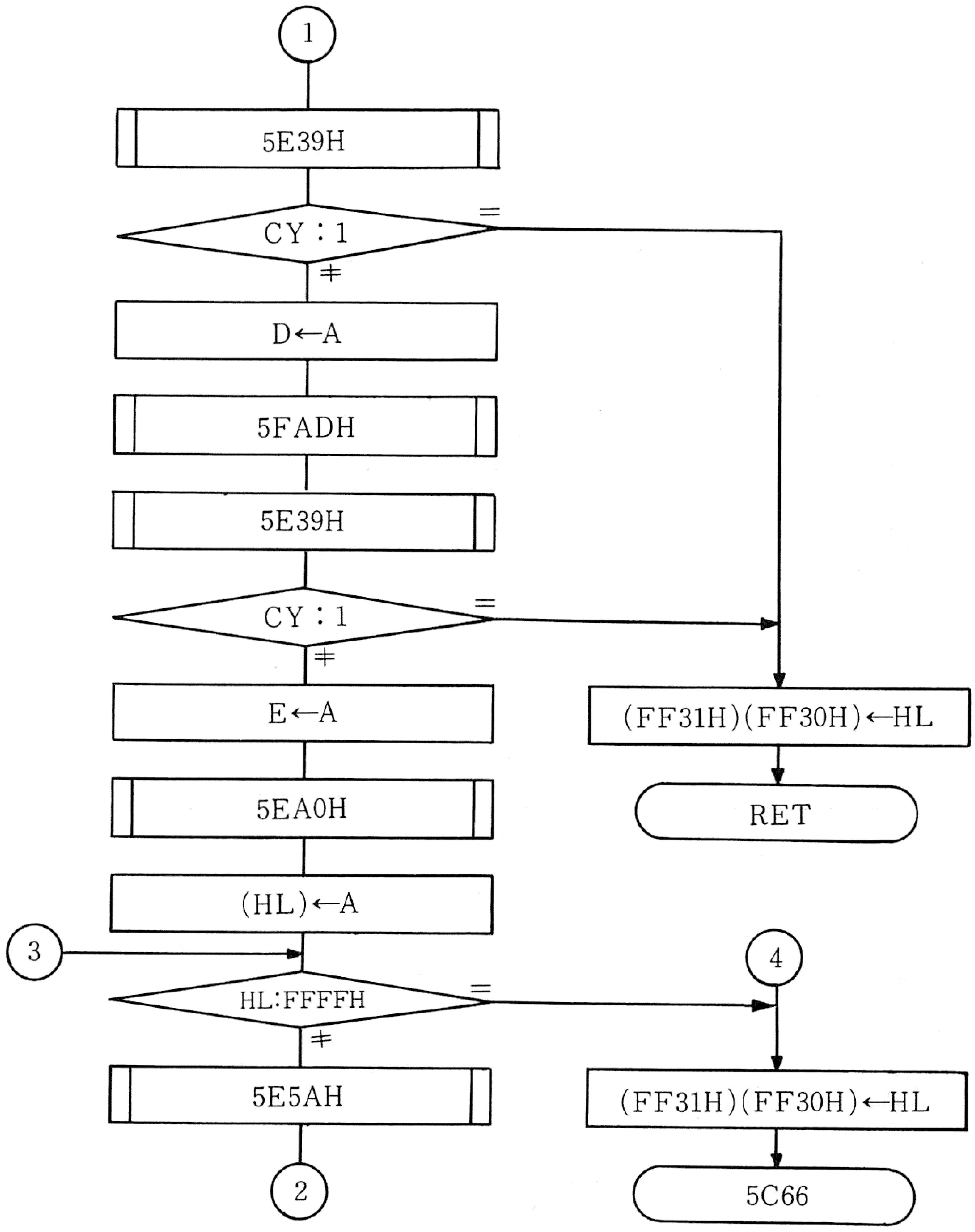
名称　マシン語モニタ　Sコマンド

レジスタ　――――

内容　マシン語モニタのセット・メモリ・コマンドのエントリ・アドレスです.

フロー・
チャート





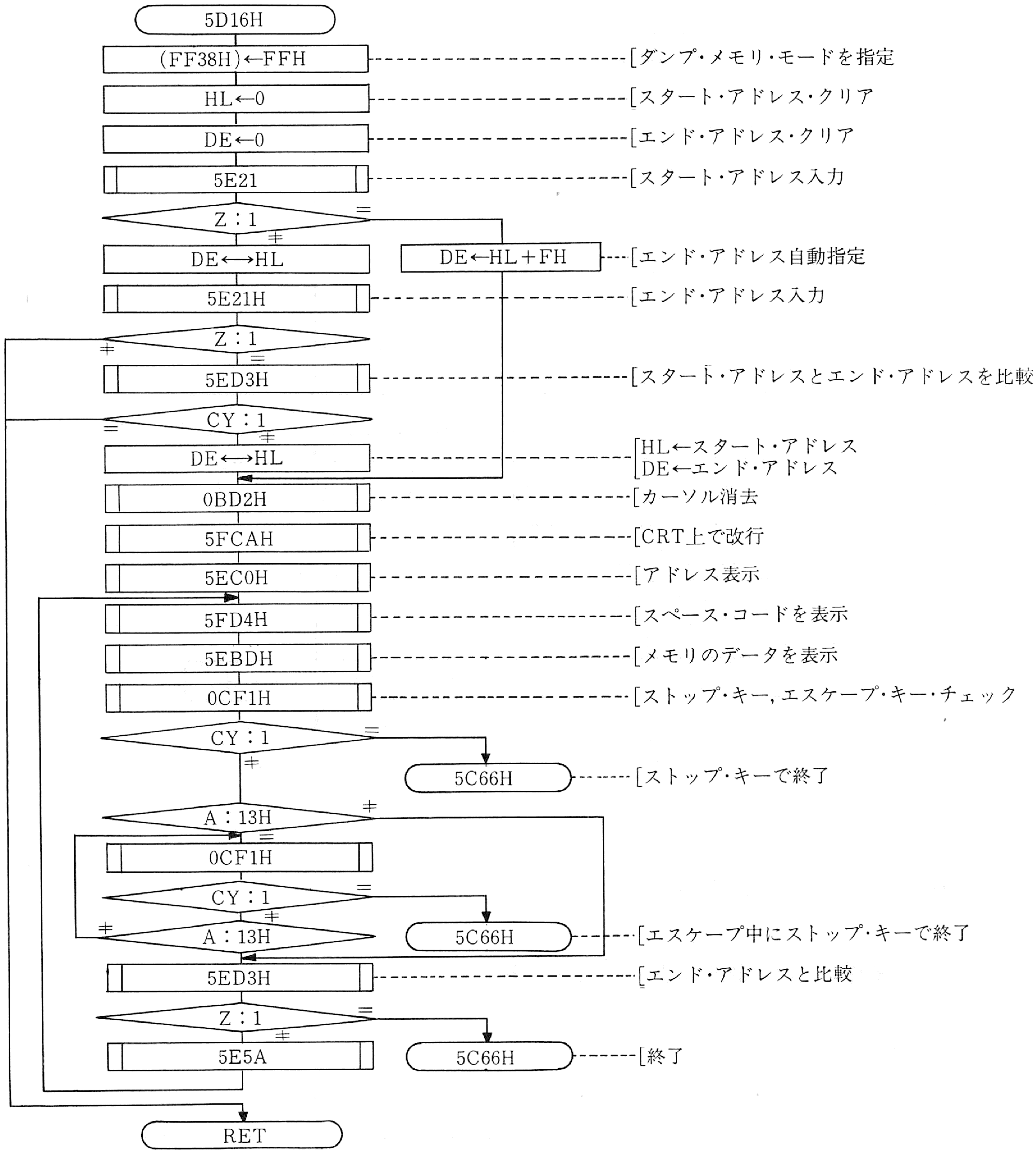
▶アドレス　5D16H

名称　マシン語モニタ　Dコマンド

レジスタ　――――

内容　マシン語モニタのダンプ・メモリ・コマンドのエントリ・アドレスです.

フロー・
チャート


5D16H
(FF38H)←FFH
[ダンプ・メモリ・モードを指定
HL←0
[スタート・アドレス・クリア
DE←0
[エンド・アドレス・クリア
5E21
[スタート・アドレス入力
Z：1
DE⟷HL
DE←HL＋FH
[エンド・アドレス自動指定
5E21H
[エンド・アドレス入力
Z：1
5ED3H
[スタート・アドレスとエンド・アドレスを比較
CY：1
DE⟷HL
[HL←スタート・アドレス
DE←エンド・アドレス
0BD2H
[カーソル消去
5FCAH
[CRT上で改行
5EC0H
[アドレス表示
5FD4H
[スペース・コードを表示
5EBDH
[メモリのデータを表示
0CF1H
[ストップ・キー，エスケープ・キー・チェック
CY：1
5C66H
[ストップ・キーで終了
A：13H
0CF1H
CY：1
A：13H
5C66H
[エスケープ中にストップ・キーで終了
5ED3H
[エンド・アドレスと比較
Z：1
5E5A
5C66H
[終了
RET

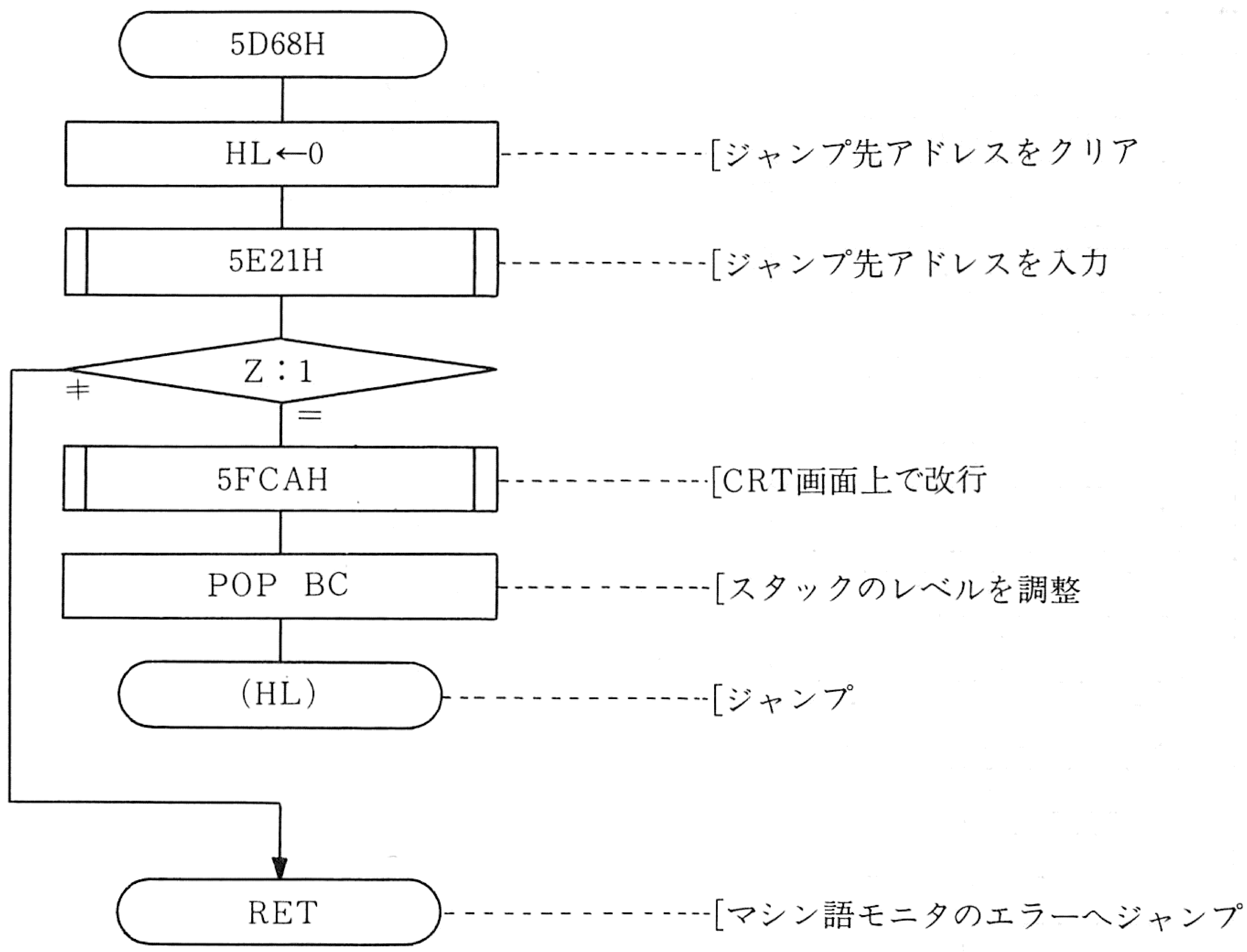
▶アドレス　5D68H

名称　マシン語モニタ　Gコマンド

レジスタ　————

内容　マシン語モニタのジャンプ・コマンドのエントリ・アドレスです.

フロー・
チャート



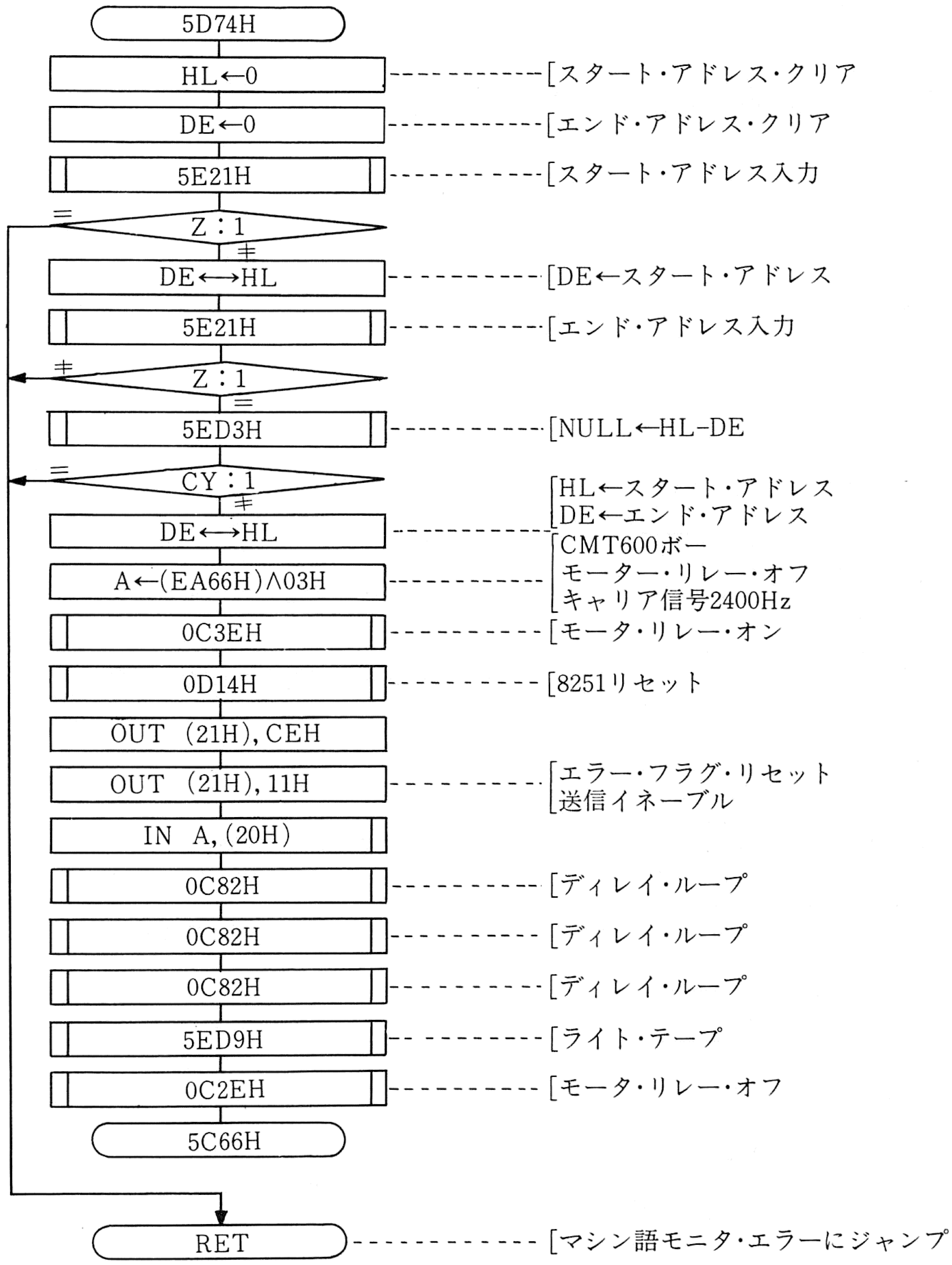
▶アドレス　5D74H

名称　マシン語モニタ　Wコマンド

レジスタ　————

内容　マシン語モニタの，ライト・テープ・コマンドのエントリ・アドレスです.

フロー・
チャート



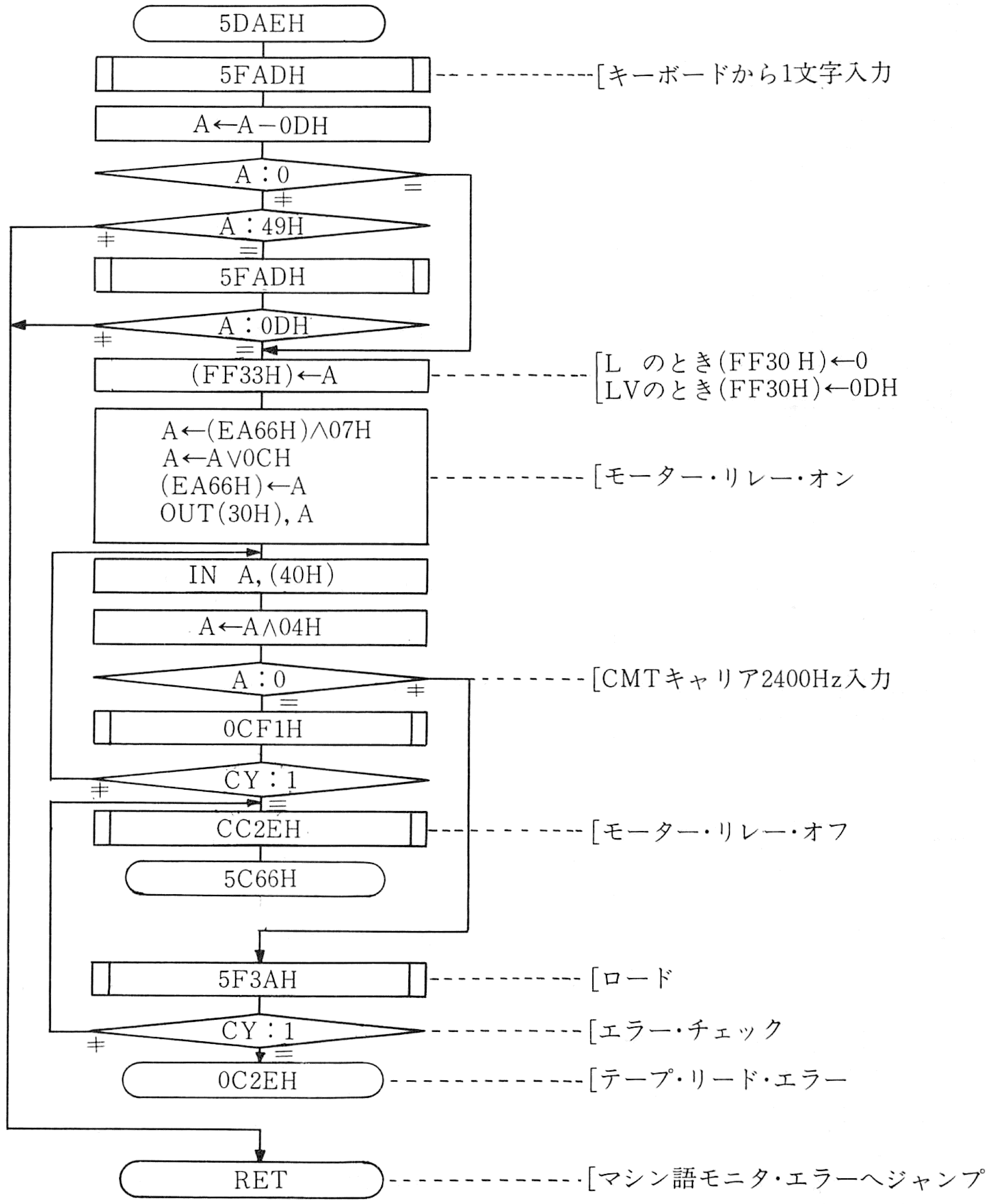
▶アドレス **5DAEH**

名称　マシン語モニタ　L，LVコマンド

レジスタ　————

内容　マシン語モニタの，ロード・テープ・コマンドおよびロード・ベリファイ・コマンドのエントリ・アドレスです．

フロー・チャート



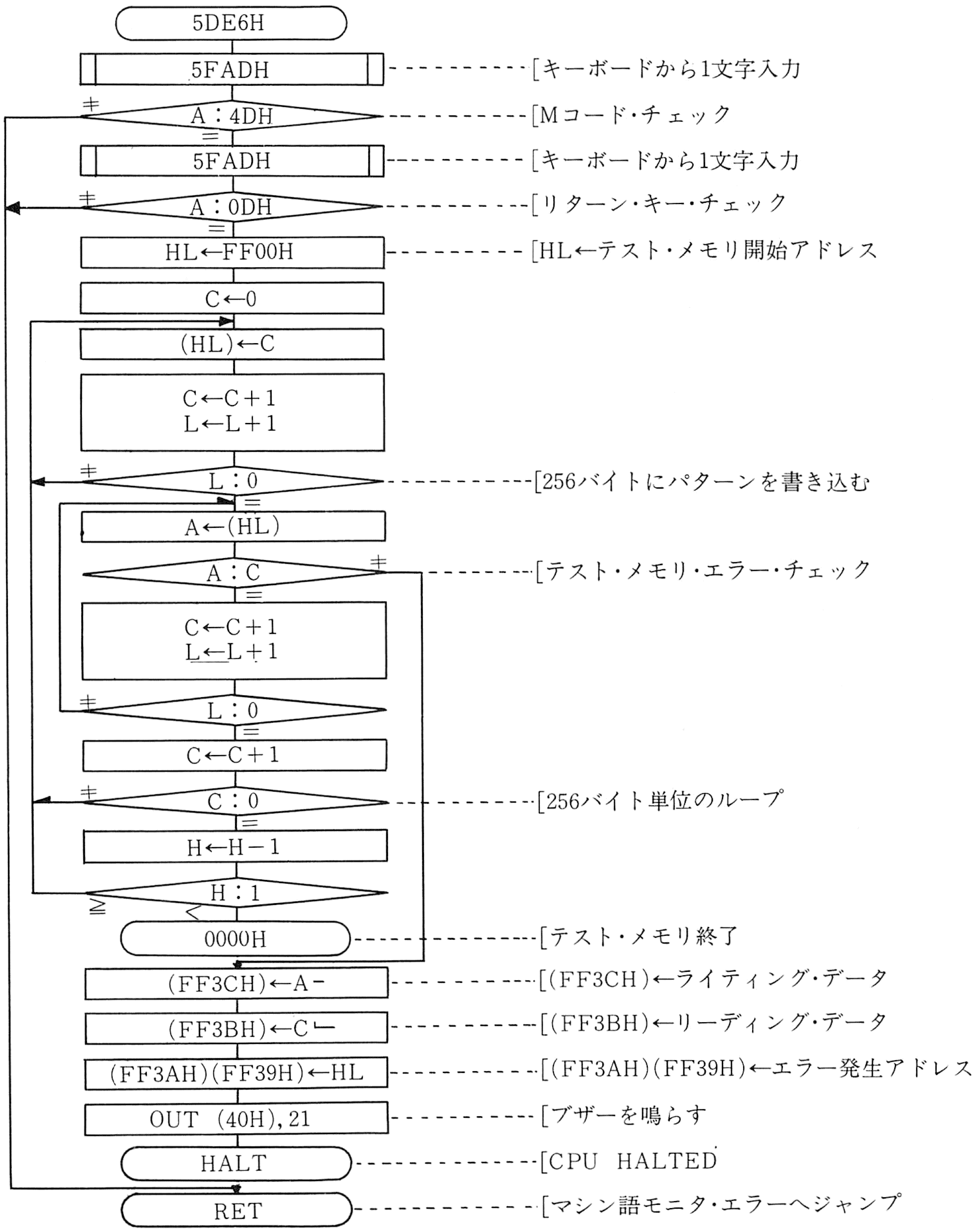
▶アドレス　**5DE6H**

名称　マシン語モニタ　TMコマンド

レジスタ　――――

内容　マシン語モニタのテスト・メモリ・コマンドのエントリ・アドレスです.

フロー・
チャート



▶アドレス　５ＤＦ２Ｈ

名称　テスト・メモリ

レジスタ　――――

内容　Ｎ－ＢＡＳＩＣのＲＡＭ領域である８０００Ｈ～ＦＦＦＦＨ番地のメモリが正常かをテストします．
この領域の全てのメモリが正常であった場合には，Ｎ－ＢＡＳＩＣのコールド・スタートを行ないますが，エラーが発生した場合には，ＨＡＬＴによって処理を停止します．停止した時点で，ＦＦ３９Ｈ～ＦＦ３ＡＨ番地に不良メモリのアドレスが，ＦＦ３ＢＨ番地に書き込んだデータ，ＦＦ３ＣＨ番地には読み出したデータが入っています．
メモリのテストは，ＲＡＭ領域の最上位バイトから２５６バイト単位で行なって行き，１６Ｋと３２Ｋシステムの区別は自動的に行なわれます．

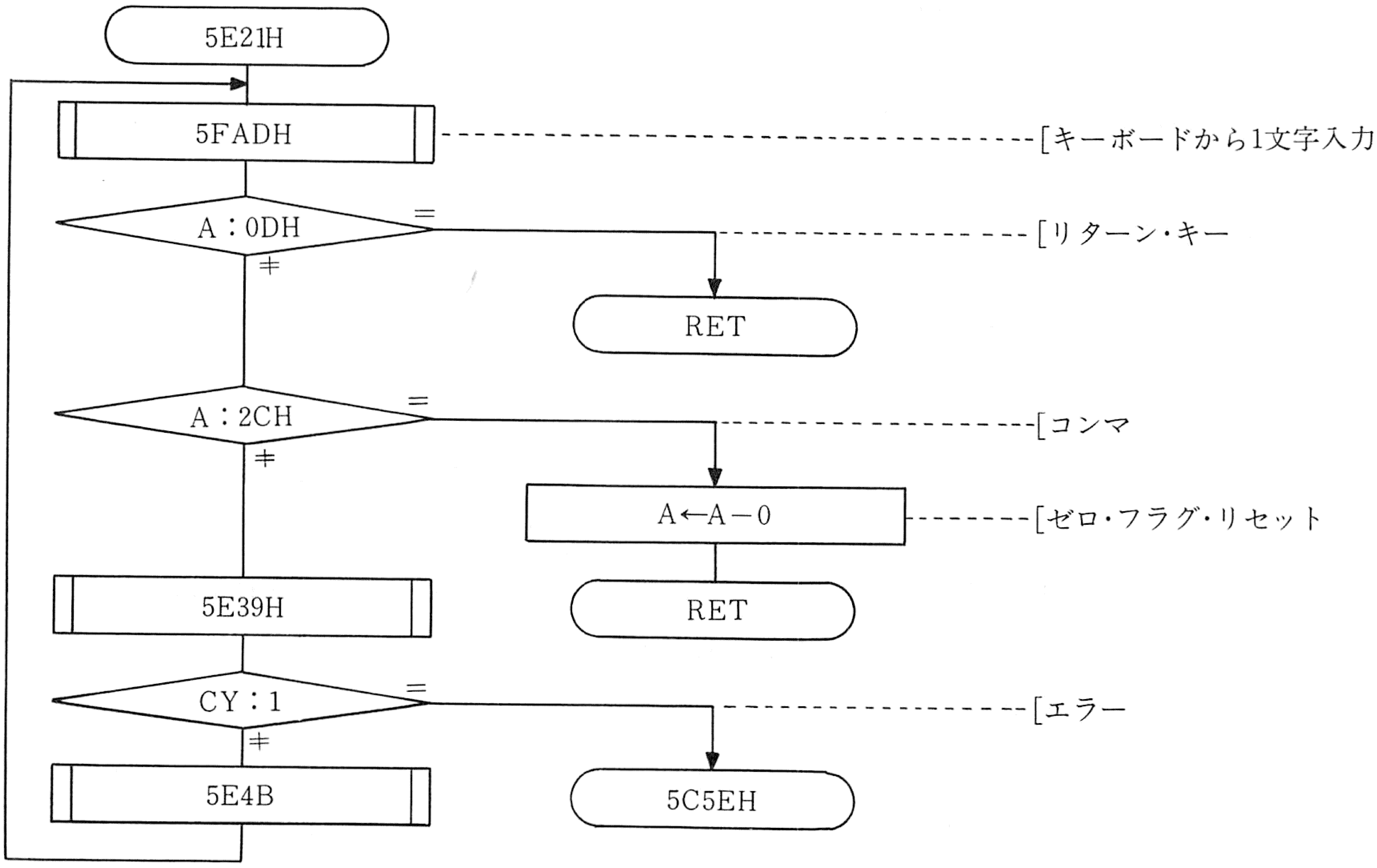
▶アドレス　５Ｅ２１Ｈ

名称　１６進４桁入力

レジスタ　Ａ，Ｆ，Ｈ，Ｌ

内容　キーボードから４桁の１６進数を入力して，ＨＬレジスタ対に格納します．
５桁以上の入力があった場合には下位４桁が有効となり，１６進関係以外のキーが入力されるとマシン語モニタのエラーに飛びます．
サブルーチンからは，リターン・キーかコンマ「，」が入力された時点でもどりますが，どちらかの区別は，リターン・キーの時にゼロ・フラグ（Ｚ）がセットされる事で区別できます．

フロー・
チャート



▶アドレス　５Ｅ３９Ｈ

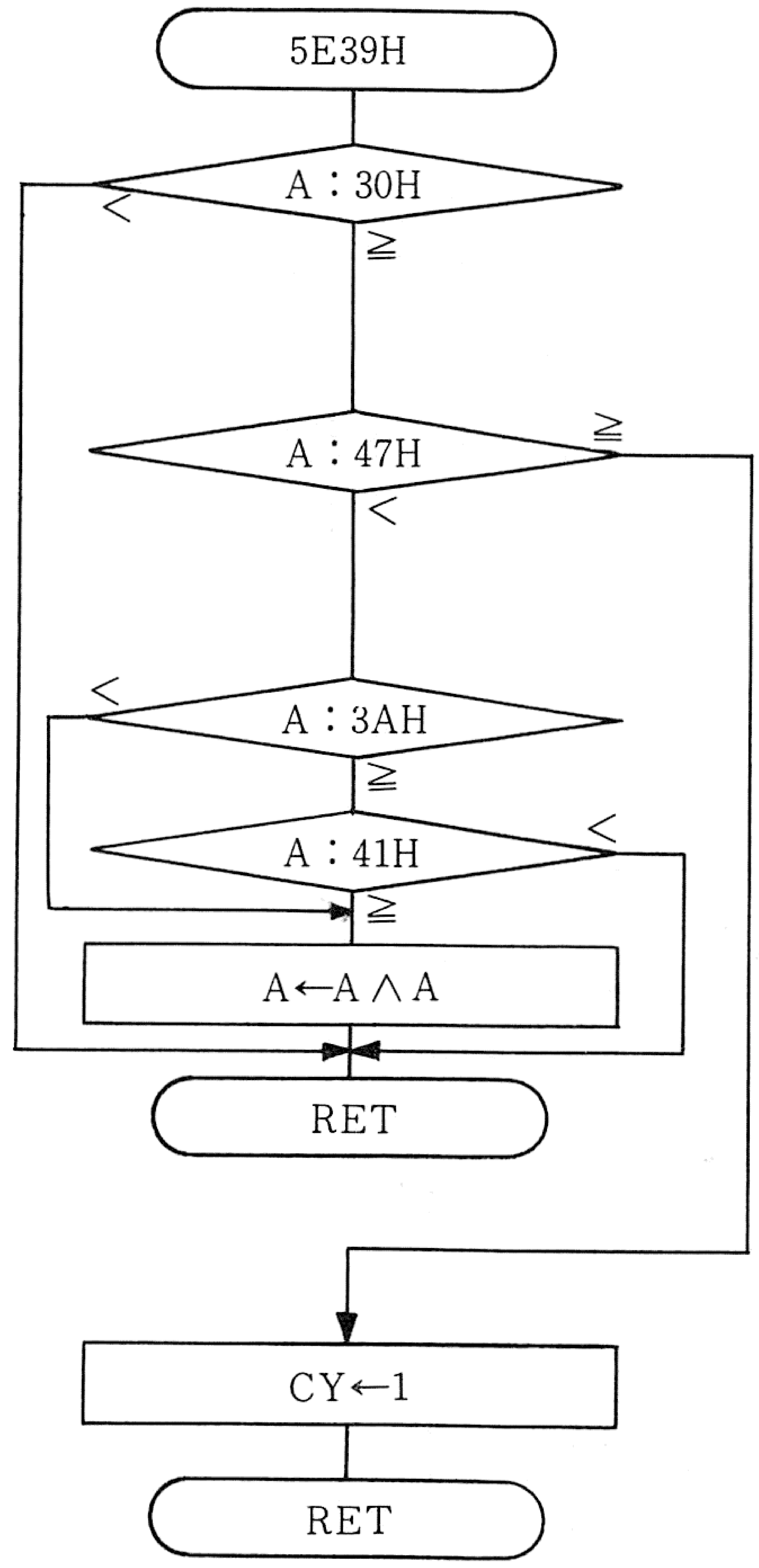
名称　１６進コード・チェック

レジスタ　Ｆ

内容　Ａレジスタのキャラクタ・コードが１６進関係でない場合には，キャリー・フラグ（ＣＹ）を１にセットします．

３０Ｈ～３９Ｈ，及び，４１Ｈ～４６Ｈのコードが１６進関係とみなされています．

フロー・チャート



▶アドレス　**5E4BH**

名称　**16進コード→バイナリ・コード変換**

レジスタ　A，F，H，L

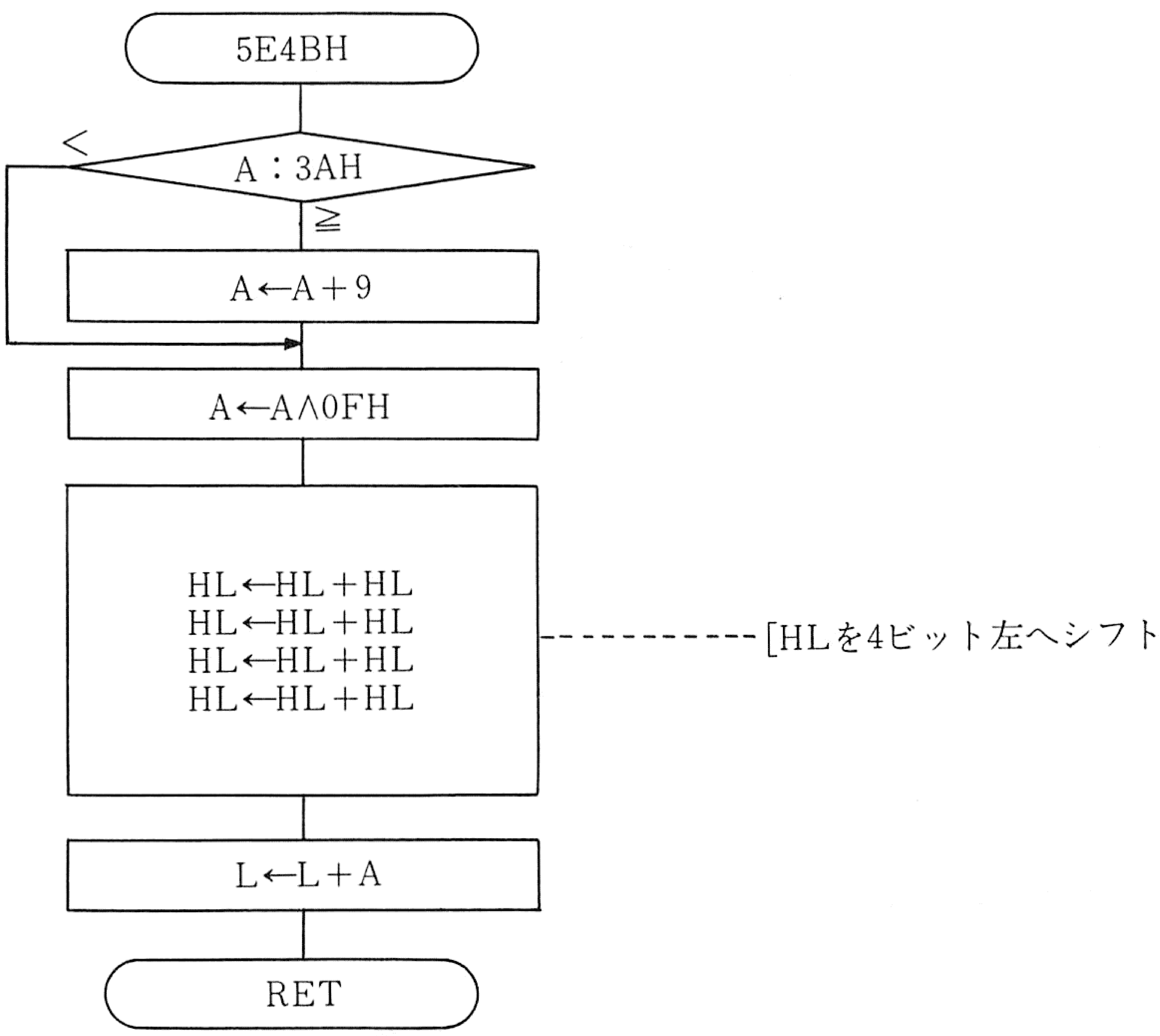
内容　HLレジスタ対の値を16倍し，Aレジスタの16進キャラクタ・コードをバイナリ・コードに変換したものを加えてもどります．
ただし，Aレジスタのコードが16進関係か否かの判定は行なっていません．

例　キーボードより16進コード1文字を入力し，数値変換したものをLレジス

タに格納する.

```
       LD    L, 0
LBL:CALL  0F75H
       CALL  5E39H
       JR    C, LBL
       CALL  5E4BH
       RET
```

フロー・
チャート



▶アドレス　5E5AH

名称　マシン語モニタ用アドレス・カウンタ・インクリメント

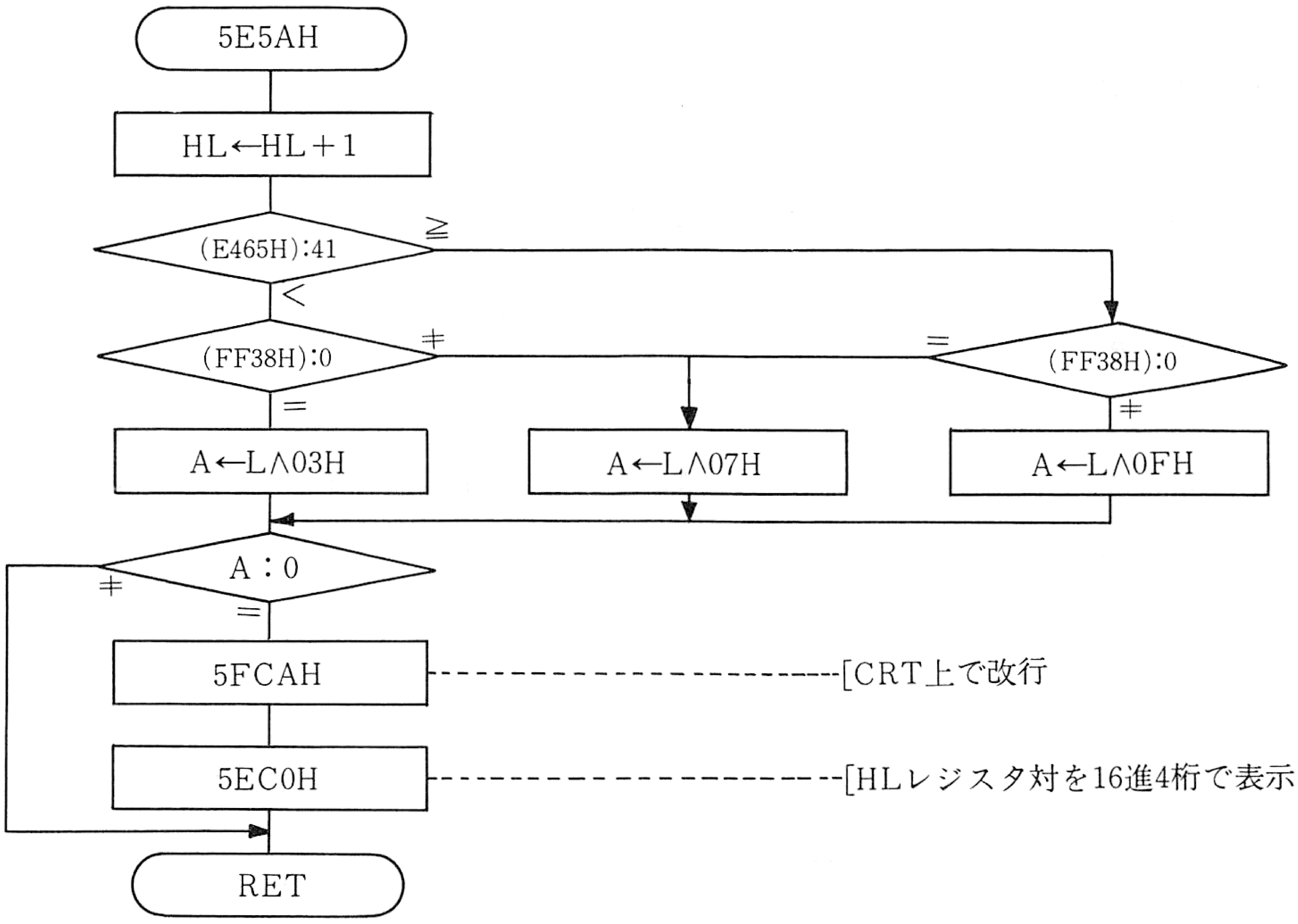
レジスタ　A, F, H, L

内容　HLレジスタ対の値をインクリメントし，必要に応じて改行の後HLレジスタ対の値を4桁の16進数としてCRT画面上に表示します.

アドレスを表示するタイミングは，セット・メモリ・コマンドかダンプ・メモリ・コマンドかによって異なりますが，これはＦＦ３８Ｈ番地によって判別します．

さらに，ＣＲＴ画面の表示桁数も自動的に判断の対象となります．

フロー・
チャート



▶アドレス　５Ｅ８３Ｈ

名称　１６進→１６進キャラクタ・コード変換

レジスタ　D，E

内容　Ａレジスタの数値データを，２バイトの１６進キャラクタ・コードに変換して，ＤＥレジスタ対に格納します．

例　Ａレジスタのデータを２桁の１６進数としてＣＲＴ画面に表示する．

```
CALL  5E83H
LD    A, D
CALL  0257H
LD    A, E
CALL  0257H
RET
```

▶アドレス　５Ｅ９６Ｈ

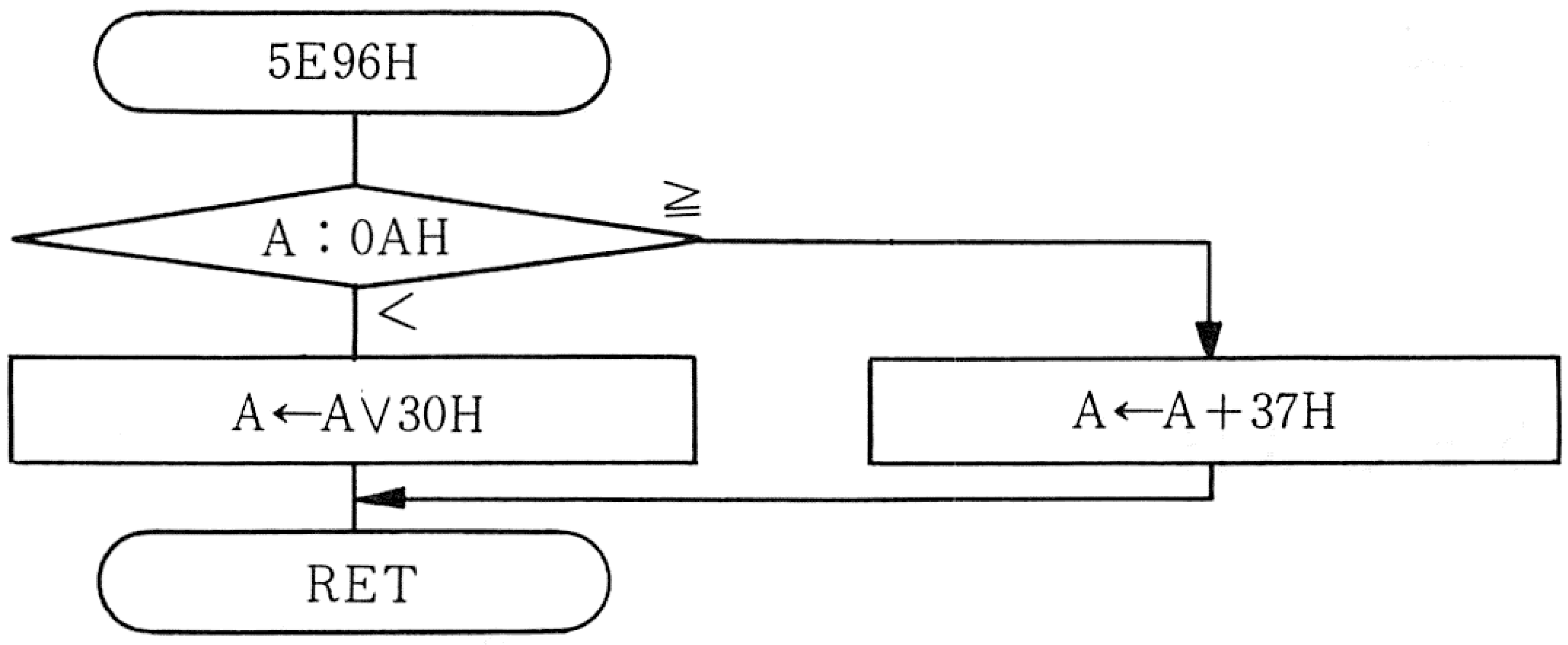
名称　４ビット１６進→コード変換

レジスタ　A, F

内容　Ａレジスタ中の００Ｈ～０ＦＨのデータを１６進のキャラクタ・コードに変換します．

ただし，入力パラメータのエラー・チェック等は行ないません．

フロー・チャート



▶アドレス　５ＥＡ０Ｈ

名称　１６進コード→１６進変換

レジスタ　A, F

内容　ＤＥレジスタ対の１６進キャラクタ・コードを，１バイトの数値データに変

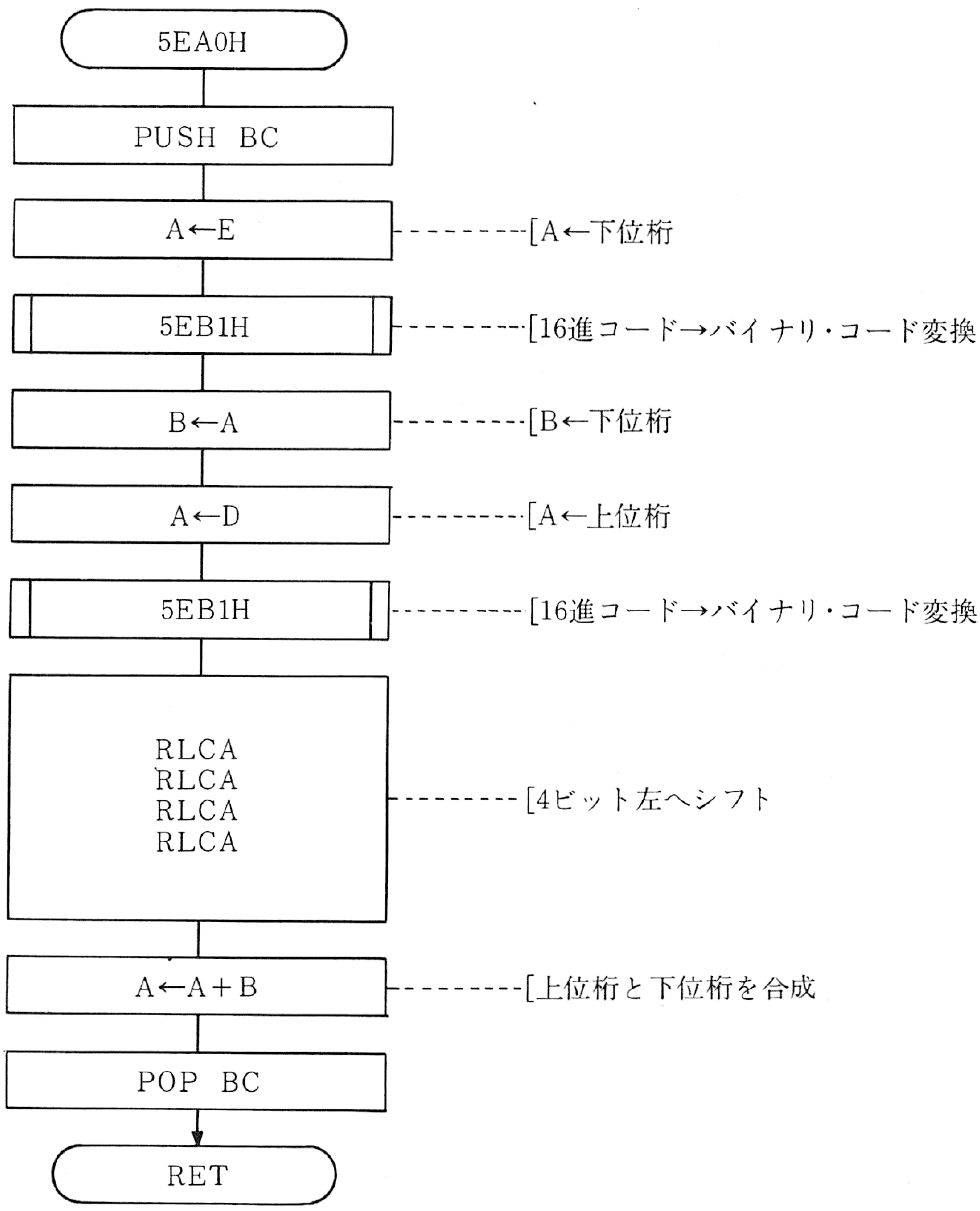
換してAレジスタに収納します.

ただし，Dレジスタと Eレジスタのコードが16進か否かの判定は行なっていません.

例　キーボードから2桁の16進数を入力し，エラー・チェックはしない.

```
CALL  0F75H
LD    D, A
CALL  0F75H
LD    E, A
CALL  5EA0H
RET
```

フロー・チャート



▶アドレス　５ＥＢ１Ｈ

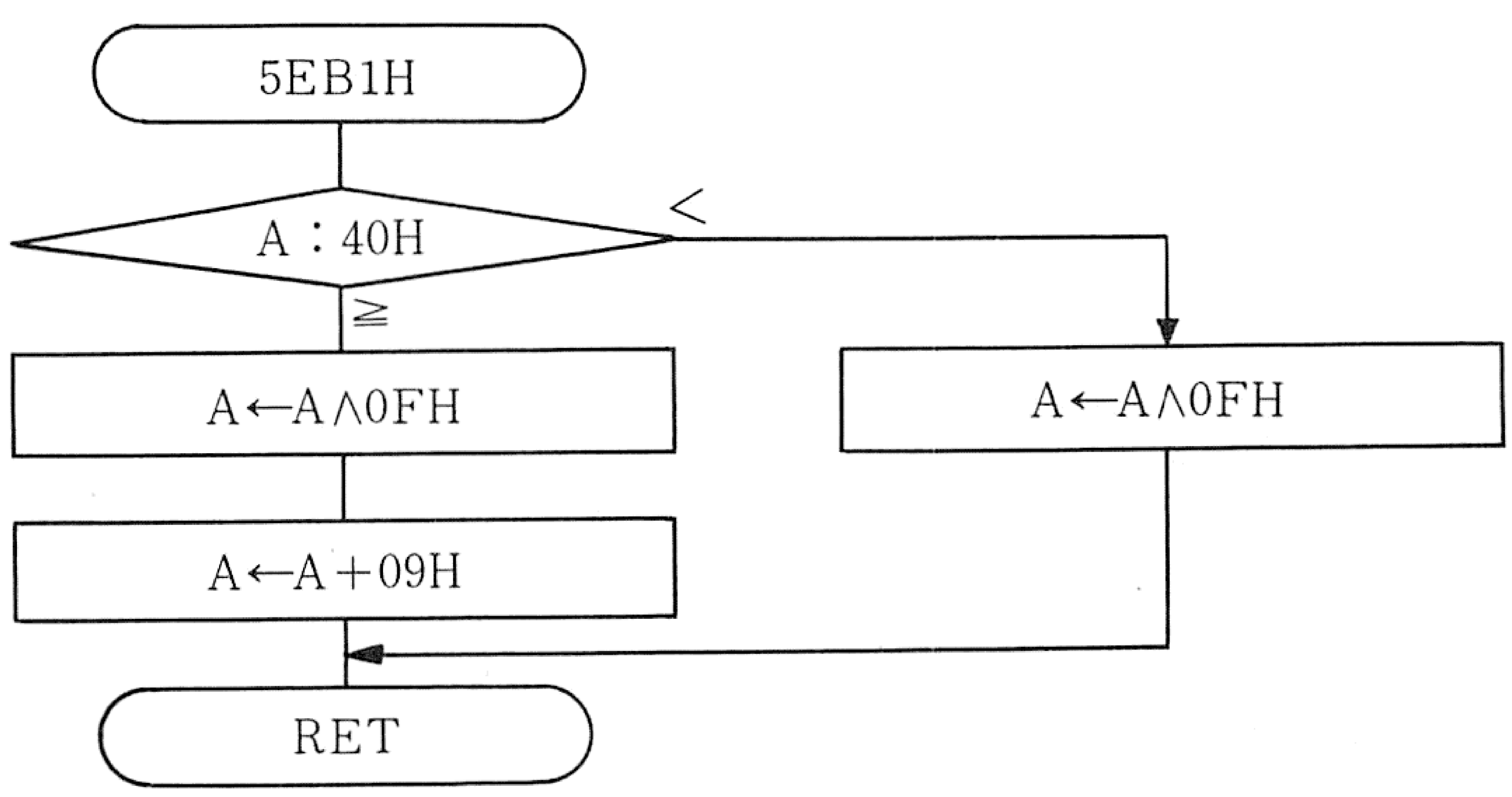
名称　１６進キャラクタ・コード→１６進数変換

レジスタ　A，F

内容　Ａレジスタの１６進キャラクタ・コードを０～ＦＨの数値データに変換します．

ただし，Ａレジスタのコードが１６進関係か否かの判定は行なっていません．

フロー・チャート



▶アドレス　５ＥＢＤＨ

名称　１６進データ２桁表示

レジスタ　A

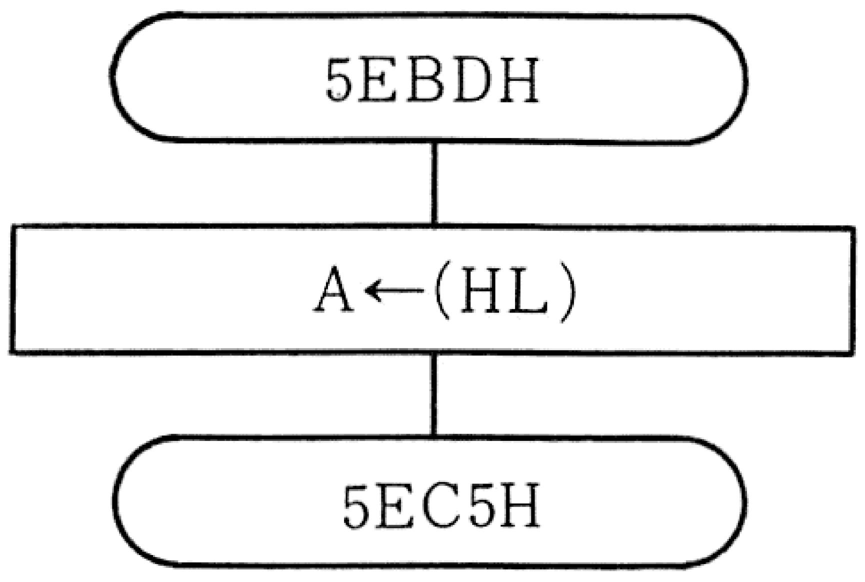
内容　ＨＬレジスタ対で指定するアドレスのデータを２桁の１６進数としてＣＲＴ画面に表示します．

例　０～ＦＦＨ番地の内容を，スペースを間にはさみながらＣＲＴ画面にダンプ

する.

```
          LD    HL, 0
LBL:CALL  5EBDH
          CALL  5FD4H
          INC   L
          JR    NZ, LBL
          RET
```

フロー・
チャート



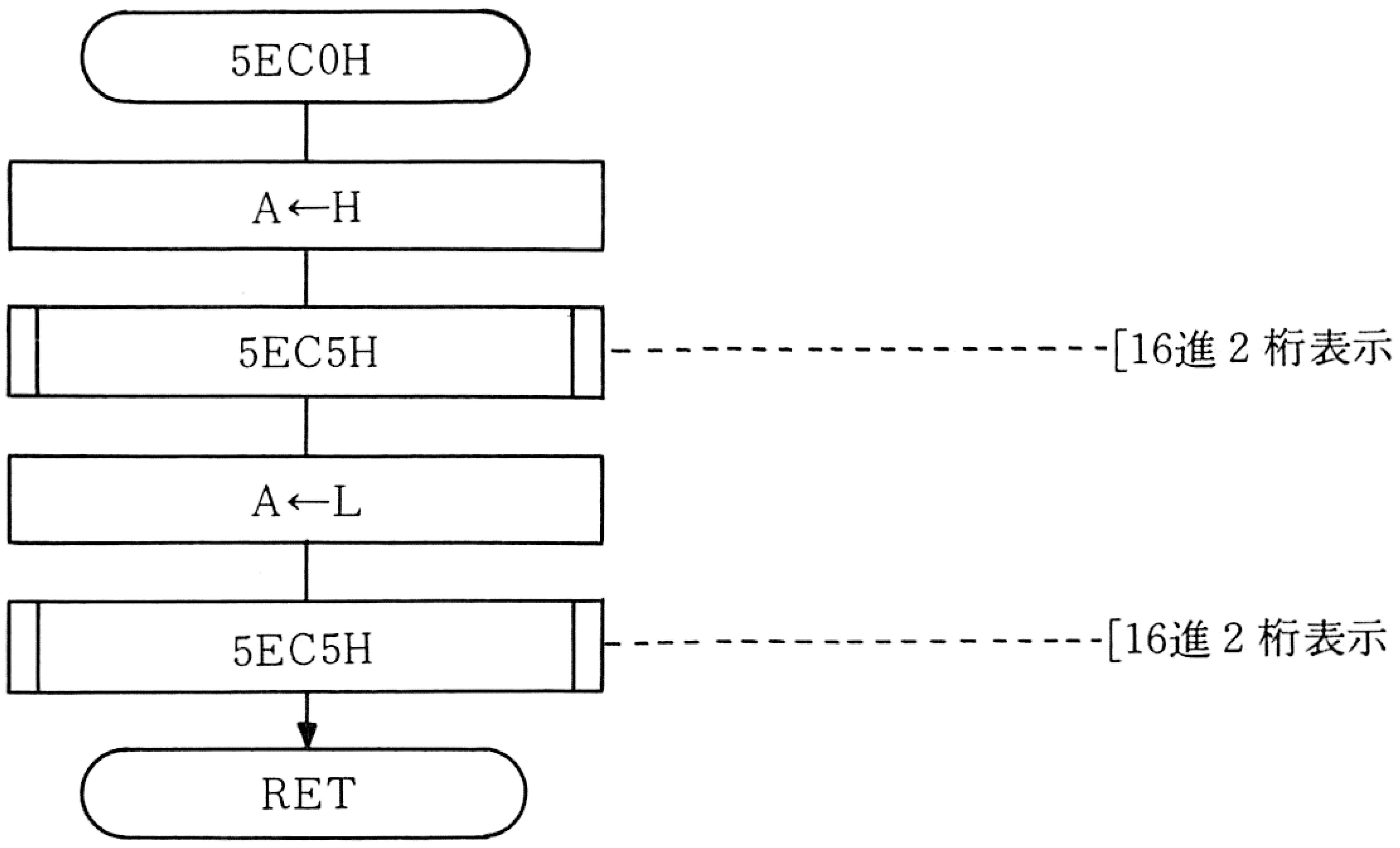
▶アドレス　5EC0H

名称　16進4桁表示

レジスタ　A

内容　HLレジスタ対の内容を，4桁の16進数としてCRT画面に表示します.

フロー・
チャート



▶アドレス　5EC5H

名称　16進2桁表示

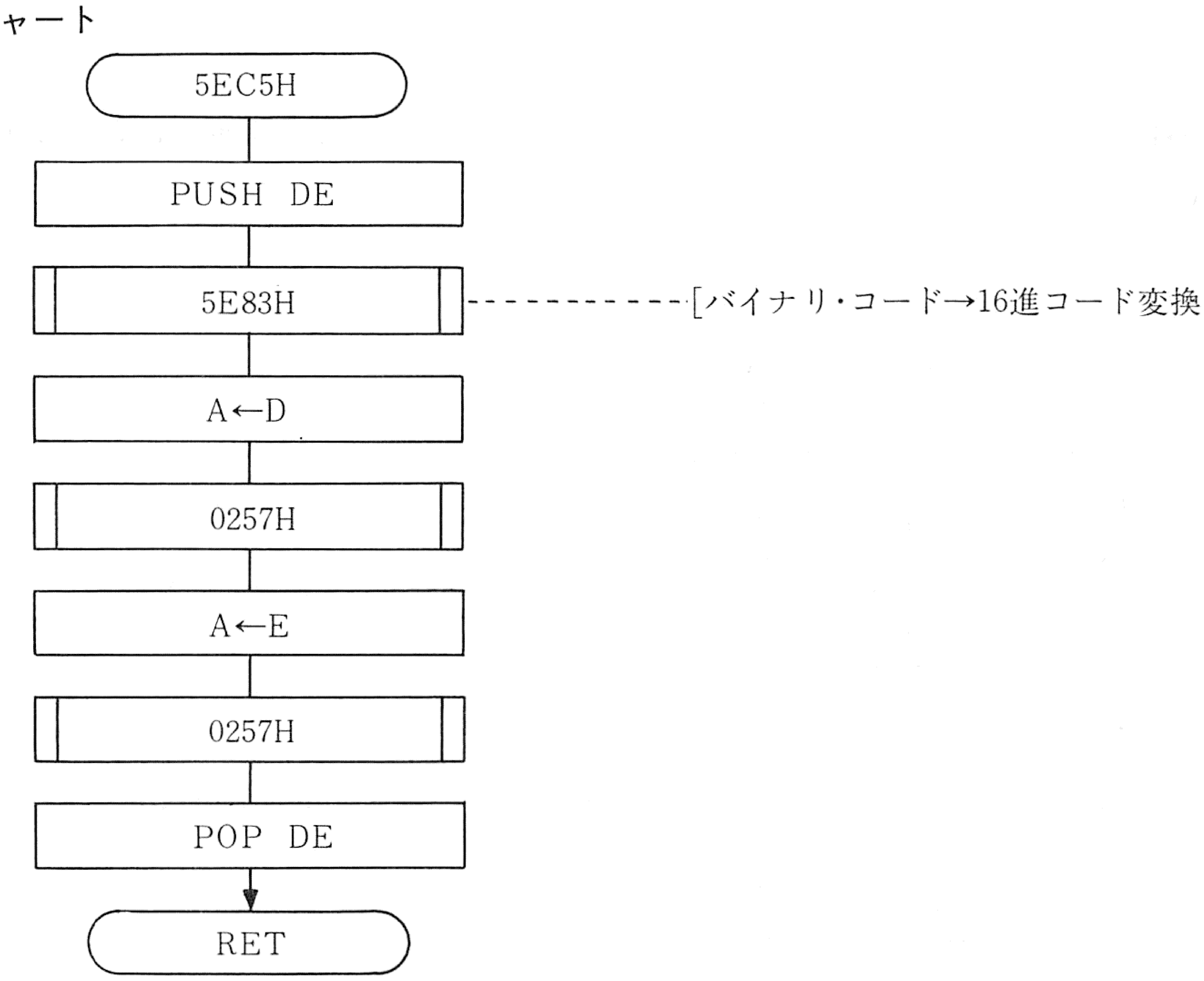
レジスタ　A

内容　Aレジスタのデータを2桁の16進数としてCRT画面に表示します.

例　0～FFHまでの16進数を改行しながらCRT画面に表示します.

```
        LD     A, 0
LBL:    PUSH   AF
        CALL   5EC5H
        CALL   5FCAH
        POP    AF
        INC    A
        JR     NZ, LBL
        RET
```

フロー・チャート



▶アドレス　５ＥＤ３Ｈ

名称　ペア・レジスタ比較

レジスタ　Ｆ

内容　ＨＬレジスタ対とＤＥレジスタ対のデータを比較します．結果は，ゼロ・フラグ（Ｚ）とキャリー・フラグ（ＣＹ）にあらわれます．

比較結果	Z	CY
ＨＬ＜ＤＥ	0	1
ＨＬ＝ＤＥ	1	0
ＨＬ＞ＤＥ	0	0

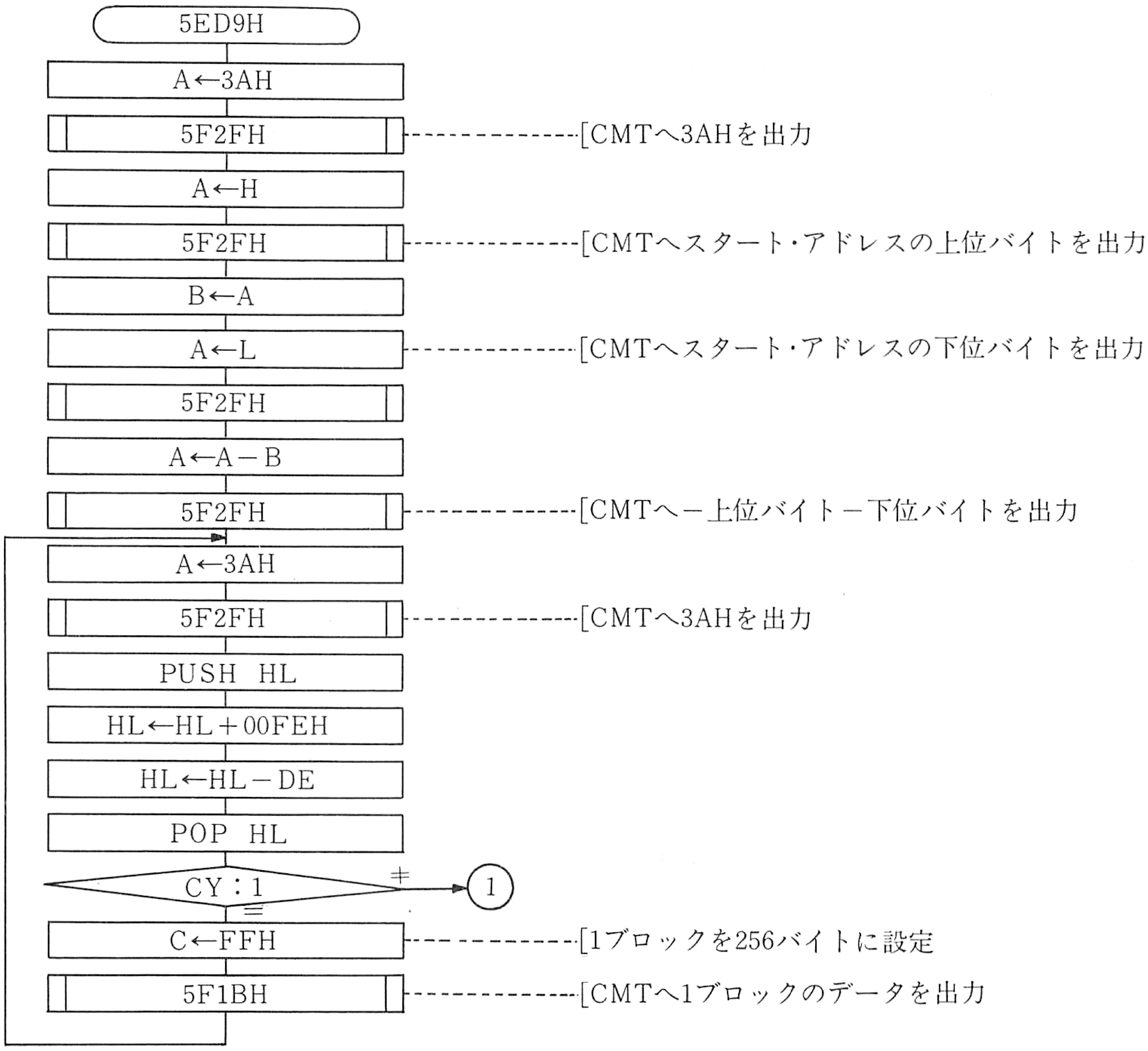
▶アドレス　５ＥＤ９Ｈ

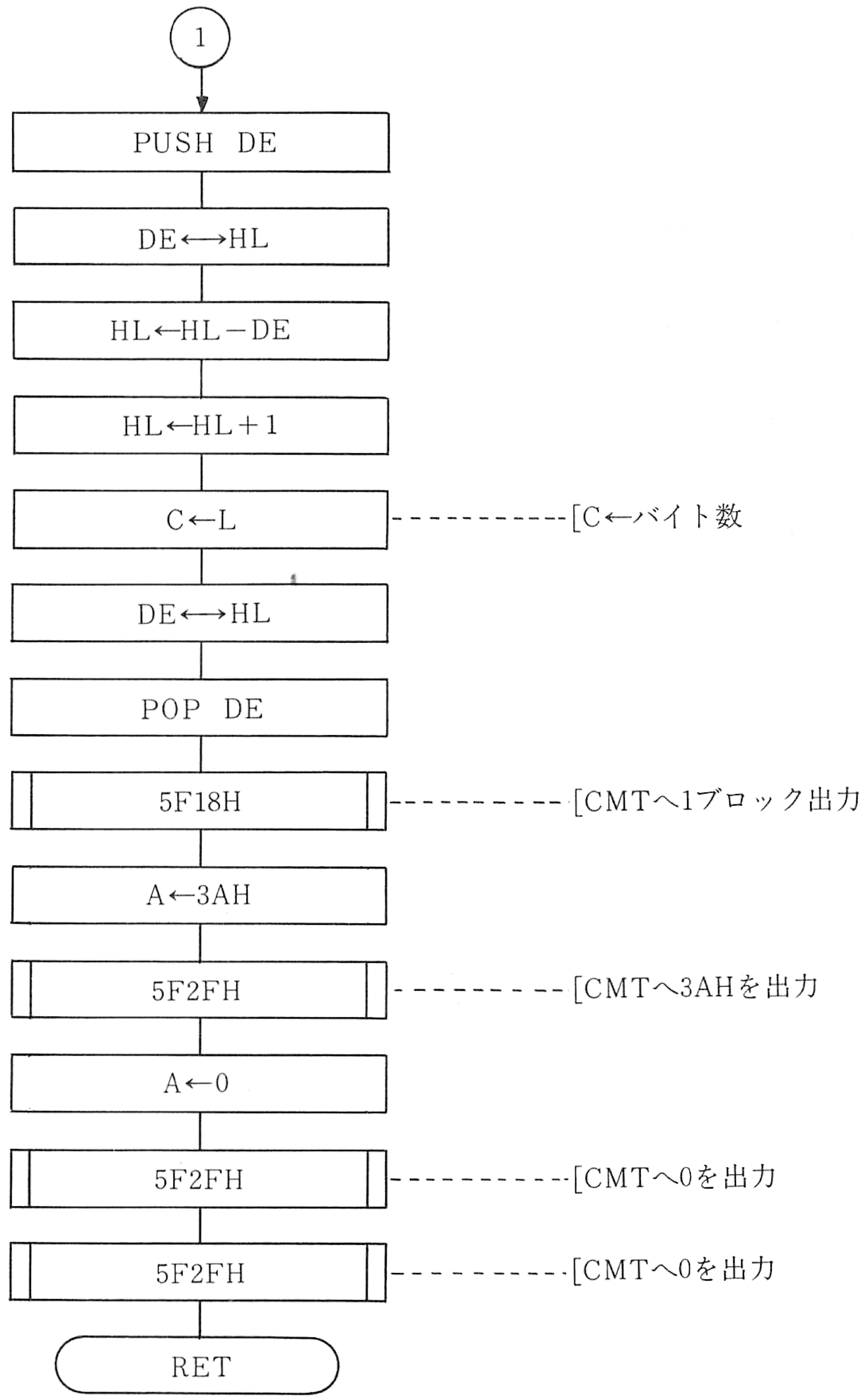
名称　ＣＭＴ出力サブルーチン

レジスタ　Ａ，Ｆ，Ｂ，Ｃ，Ｈ，Ｌ

内容　マシン語モニタのライト・テープ・コマンドからコールされるデータ出力サブルーチンです．

フロー・
チャート


5ED9H
A←3AH
5F2FH
[CMTへ3AHを出力
A←H
5F2FH
[CMTへスタート・アドレスの上位バイトを出力
B←A
A←L
[CMTへスタート・アドレスの下位バイトを出力
5F2FH
A←A−B
5F2FH
[CMTへ−上位バイト−下位バイトを出力
A←3AH
5F2FH
[CMTへ3AHを出力
PUSH HL
HL←HL＋00FEH
HL←HL−DE
POP HL
CY：1
≠
＝
1
C←FFH
[1ブロックを256バイトに設定
5F1BH
[CMTへ1ブロックのデータを出力



▶アドレス　5EEDH

名称　CMTデータ出力

レジスタ　A，F，B，C，H，L

内容　HLレジスタ対で示すアドレスからDEレジスタ対で示すアドレスまでのメモリの内容を，CMTに出力します．

出力は，マシン語モニタのフォーマットにて行なわれ，２５６バイト単位でのチェックも行ないます．

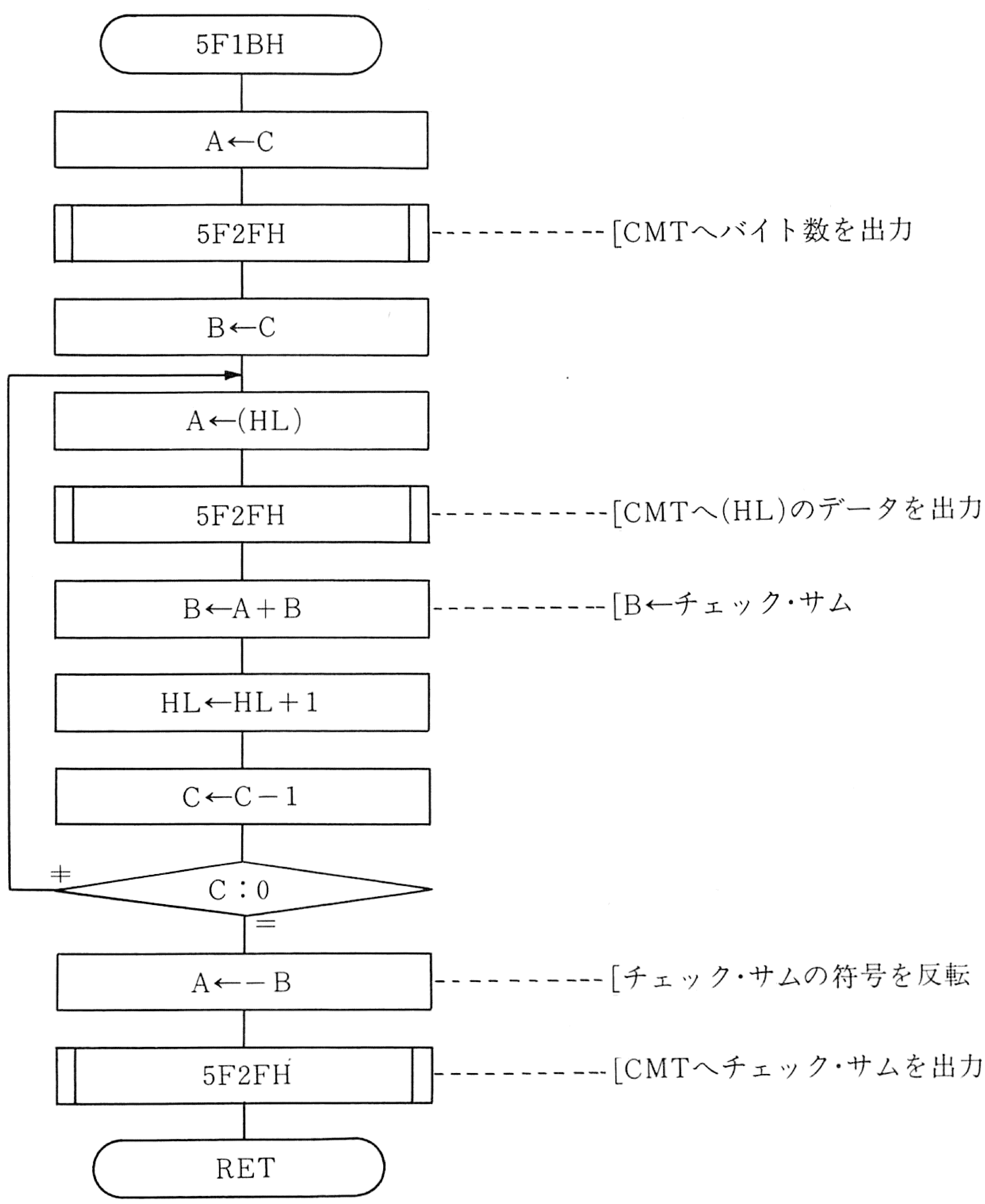
▶アドレス　５Ｆ１ＢＨ

名称　ＣＭＴデータ・ブロック出力

レジスタ　————

内容　ＣＭＴへＣレジスタが示すバイト数のデータおよびチェック・サムを出力するサブルーチンで，マシン語モニタのデータ出力ルーチンからコールされます．

フロー・チャート



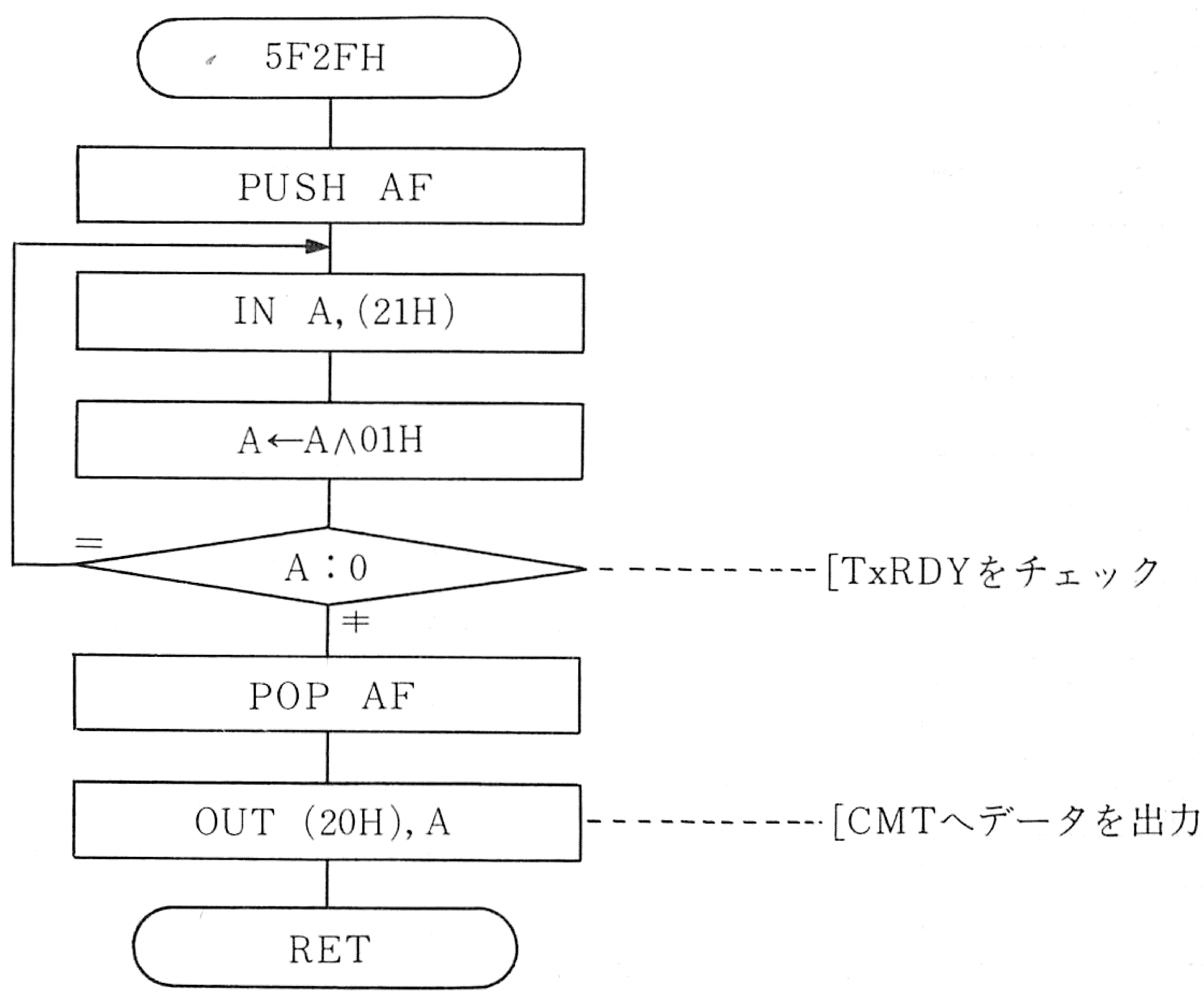
▶アドレス　5F2FH

名称　CMT 1バイト出力

レジスタ　全て保存

内容　Aレジスタのデータ1バイトをCMTに出力してもどります.

フロー・チャート



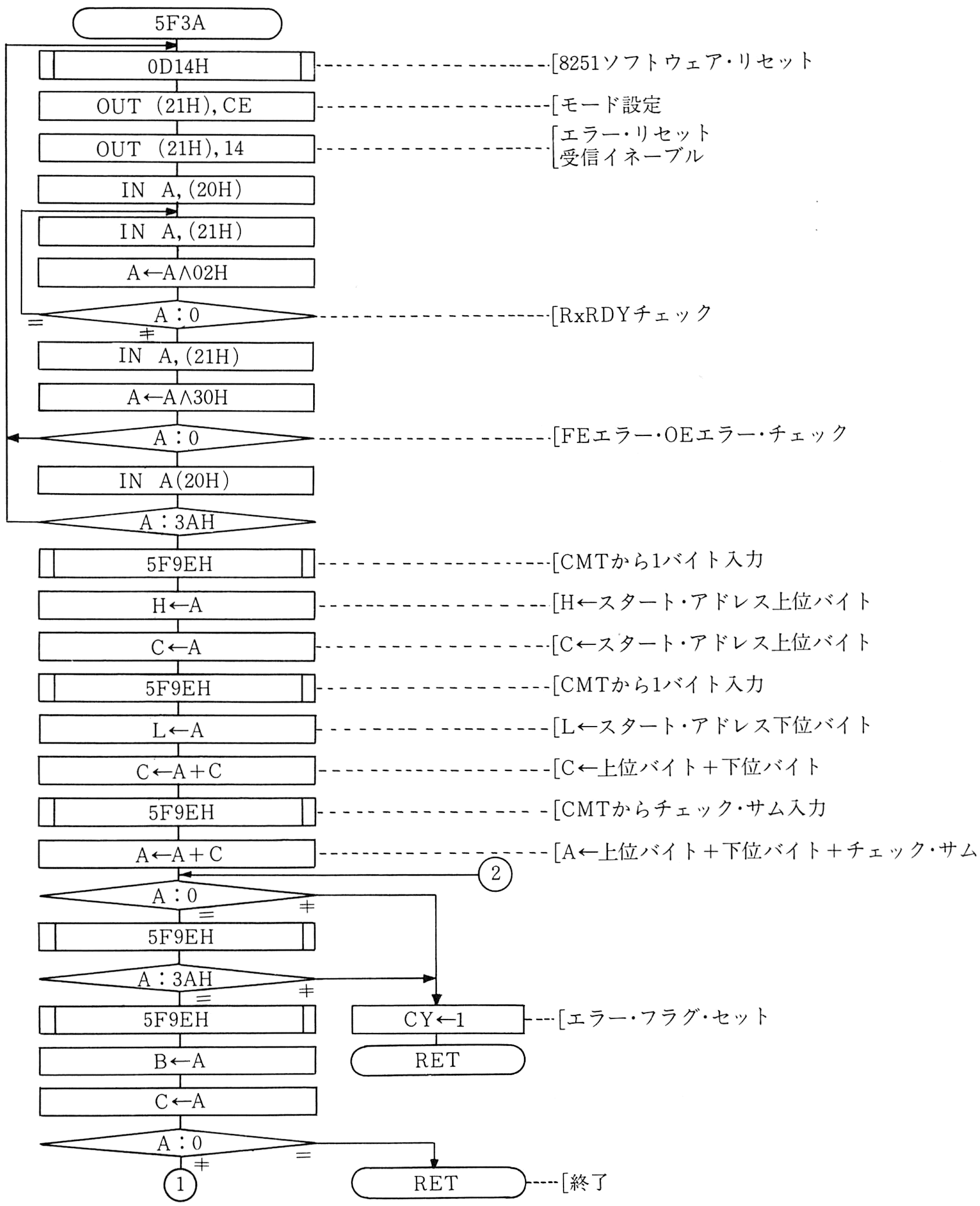
▶アドレス　5F3AH

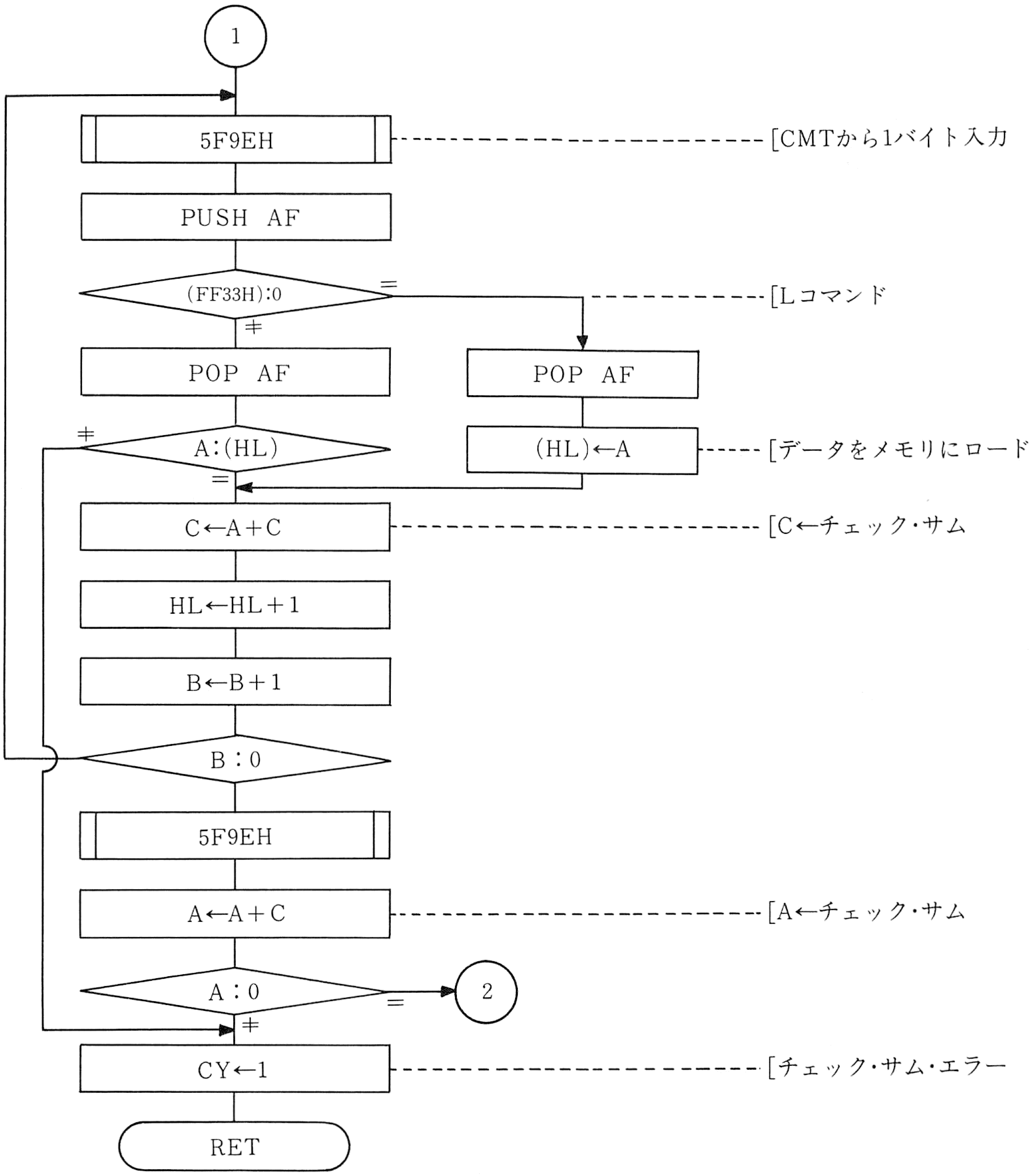
名称　CMT入力サブルーチン

レジスタ　A, F, B, C, H, L

内容　マシン語モニタのロードおよびベリファイ・コマンドからコールされるデータ入力サブルーチンです.

フロー・
チャート


5F3A
0D14H
[8251ソフトウェア・リセット
OUT (21H), CE
[モード設定
OUT (21H), 14
[エラー・リセット
受信イネーブル
IN A, (20H)
IN A, (21H)
A←A∧02H
A : 0
[RxRDYチェック
IN A, (21H)
A←A∧30H
A : 0
[FEエラー・OEエラー・チェック
IN A(20H)
A : 3AH
5F9EH
[CMTから1バイト入力
H←A
[H←スタート・アドレス上位バイト
C←A
[C←スタート・アドレス上位バイト
5F9EH
[CMTから1バイト入力
L←A
[L←スタート・アドレス下位バイト
C←A+C
[C←上位バイト+下位バイト
5F9EH
[CMTからチェック・サム入力
A←A+C
[A←上位バイト+下位バイト+チェック・サム
2
A : 0
5F9EH
A : 3AH
5F9EH
CY←1
[エラー・フラグ・セット
RET
B←A
C←A
A : 0
1
RET
[終了



▶アドレス　5F6AH

名称　CMTデータ入力

レジスタ　A, F, B, C, D, E, H, L

内容　マシン語モニタのフォーマットによりセーブされたデータをCMTから入力

し，HLレジスタ対で指定したアドレス以降のメモリに収納します.

▶アドレス　5F9EH

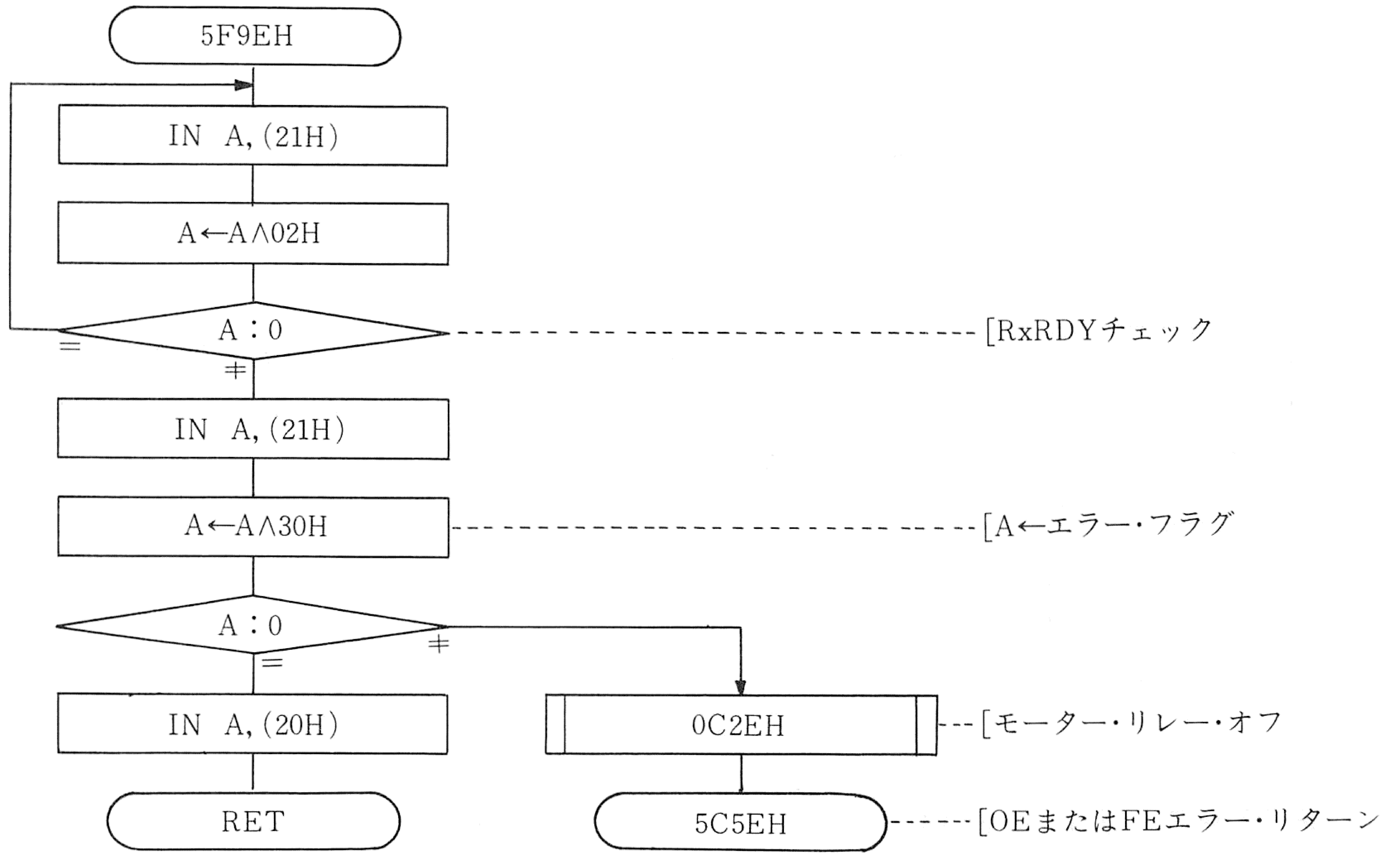
名称　CMT1バイト入力

レジスタ　A，F

内容　CMTから1バイトのデータをAレジスタに入力してもどります.

エラーが発生した場合には，マシン語モニタのエラー・ルーチンに入ります.

フロー・チャート



▶アドレス　5FADH

名称　1文字入力と英小文字→英大文字変換

レジスタ　A，F

内容　カーソルを表示してキー入力を待ち，入力されたコードが英小文字の場合には英大文字に変換します.

入力されたコードはAレジスタに入れてもどり，同時にCRT画面にもエコー・バックします.

ただし，ストップ・キーが押されるとマシン語モニタのコマンド待ちである5C66H番地にジャンプします.

▶アドレス　5FB0H

名称　カーソル表示と1文字表示

レジスタ　全て保存

内容　カーソルを表示して，Aレジスタのキャラクタ1文字をCRT画面上のカーソル位置に表示します.

▶アドレス　5FB9H

名称　1文字入力と英小文字→英大文字変換

レジスタ　A，F

内容　キー入力を待ち，入力されたコードが英小文字の場合には英大文字に変換します.

入力されたコードはAレジスタに入れてもどりますが，CRT画面へのエコー・バックは行ないません.

ただし，ストップ・キーが押されるとマシン語モニタのコマンド待ちである5C66H番地にジャンプします.

▶アドレス　5FC1H

名称　英小文字→英大文字変換

レジスタ　A, F

内容　Aレジスタのキャラクタ・コードが英小文字であれば，第5ビットをマスクして，英大文字のコードに変換します．

例　N－BASIC入力バッファ内の全英小文字を英大文字に変換する．

```
        LD    HL, EC95H
        LD    B, FFH
LBL: INC   HL
        LD    A, (HL)
        CALL  5FC1H
        LD    (HL), A
        DJNZ  LBL
        RET
```

▶アドレス　5FCAH

名称　CRT改行

レジスタ　A

内容　CRT画面に，キャリッジ・リターン・コード（0DH）とライン・フィード・コード（0AH）を出力します．
カーソル位置は現在の次の行の先頭に移動し，もしカーソルがスクロール範囲の最下行にあれば，スクロール・アップを行ないます．

▶アドレス　５ＦＤ４Ｈ

名称　スペース表示

レジスタ　Ａ

内容　ＣＲＴ画面上のカーソル位置に，スペース･コード（２０Ｈ）を表示します．

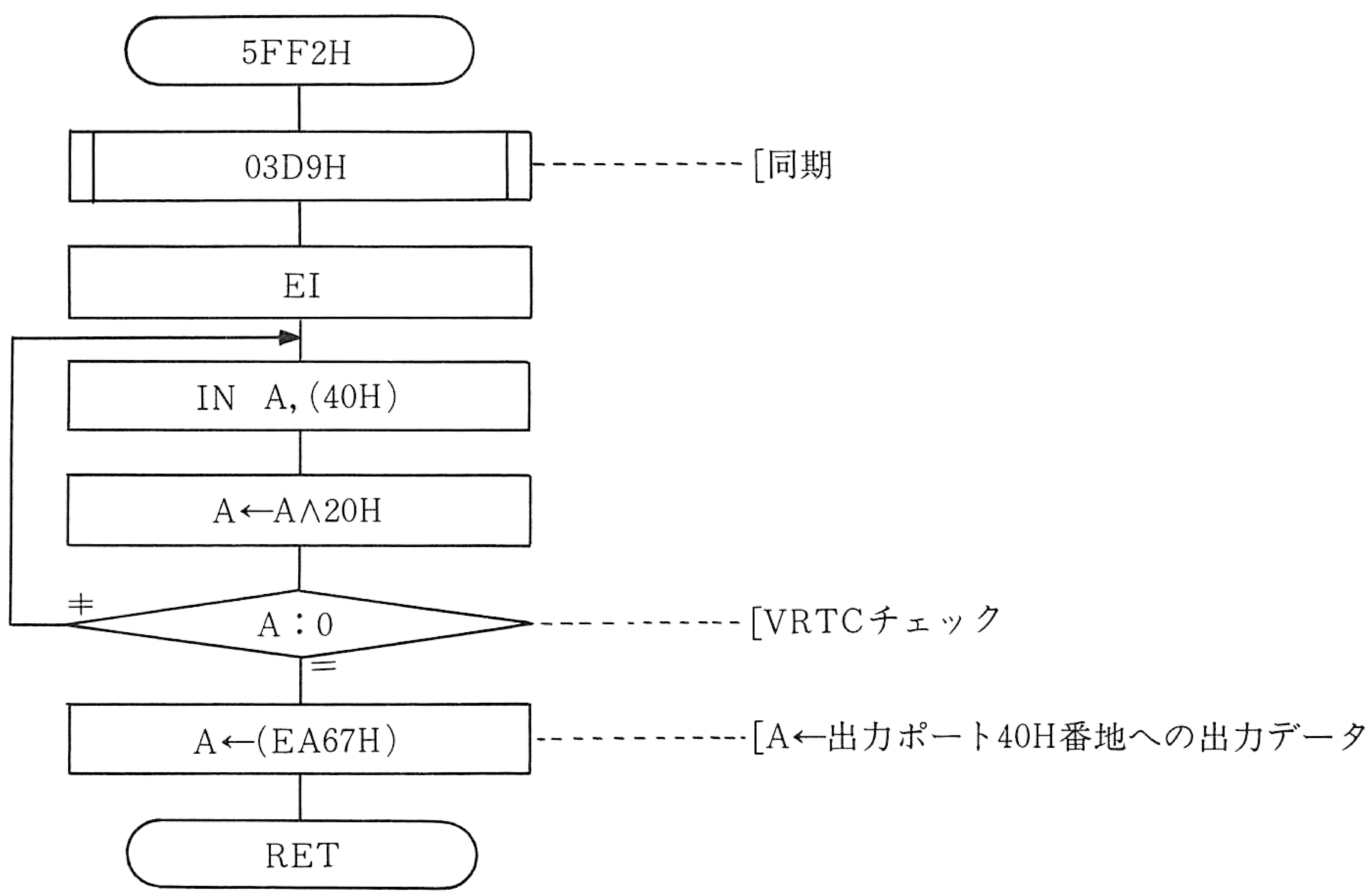
▶アドレス　５ＦＦ２Ｈ

名称　ＶＲＴＣチェック

レジスタ　Ａ，Ｆ

内容　ＶＲＴＣをチェックし，バーチカル・リトレース期間から，表示およびホリゾンタル・リトレース期間まで待ち，直前の出力ポート４０Ｈ番地への出力データをＡレジスタに入れてもどります．

フロー・チャート



1.1
Disk version [14-Nov-1980]
システム・ワーキング・エリア

E９D４H　同時にオープンできるファイルの数で，Disk－BASIC起動時に入力した値。

E９D５H～E９D６H　バッファ・ポインタの開始アドレス.

E９D７H～E９D８H　FATポインタの開始アドレス.

E９D９H～E９DAH　ファイルナンバ０用バッファの先頭アドレス.

E９DBH～E９DCH　直前にアクセスしたドライブ内のFATのメッセージ・エリア内のトラック・ナンバを示すアドレス.

E９DDH～E９DEH　ファイルナンバ０用バッファのポインタ.

E９E０H～E９E１H　直前にロードまたはセーブを行なったドライブ・ナンバ.

E９E２H～E９EAH　直前にロードまたはセーブを行なったファイル・ネーム.

E９EBH～E９F３H　NAMEで使用するファイル・ネーム.

E９F４H　DSKO＄のトラック・ナンバ.

E９F５H　DSKO＄のセクタ・ナンバ.

1.2
N-BASIC
システム・ワーキング・エリア

ＥＡ００Ｈ～ＥＡ０２Ｈ	ＥＡ０１Ｈ番地にＩ／Ｏポートのアドレスを入れてコールすると，Ａレジスタのデータを出力するサブルーチンで，Ｎ－ＢＡＳＩＣのＯＵＴ等で使用.
ＥＡ０３Ｈ～ＥＡ１０Ｈ	ＥＡ０４Ｈ，ＥＡ０８Ｈ，ＥＡ０ＣＨ，ＥＡ０ＦＨ番地に数値を代入してから用いる単精度減算用サブルーチンで，除算等で使用.
ＥＡ１１Ｈ	乱数用．乱数＝Ｆ（前の乱数＊Ｘ＋Ｙ）のＦに関する値（０～ＡＢＨ).
ＥＡ１２Ｈ	乱数用．下位２ビットで３種類のＹを選択（０～ＦＦＨ).
ＥＡ１３Ｈ	乱数用．下位３ビットで８種類のＸを選択（０～ＦＦＨ).
ＥＡ１４Ｈ～ＥＡ３３Ｈ	単精度の乱数用定数８種類.
ＥＡ３４Ｈ～ＥＡ３７Ｈ	前に発生した乱数の退避用.
ＥＡ３８Ｈ～ＥＡ３ＡＨ	ＥＡ３９Ｈ番地にＩ／Ｏポートのアドレスを入れてコールすると入力データをＡレジスタに入れるサブルーチンで，Ｎ－ＢＡＳＩＣのＩＮ等で使用.
ＥＡ３ＢＨ～ＥＡ４ＥＨ	ＵＳＲ関数，ＵＳＲ０～ＵＳＲ９のコール先アドレスで，初期値は４４Ａ５Ｈ.
ＥＡ４ＦＨ	未使用.
ＥＡ５０Ｈ～ＥＡ５１Ｈ	フリー・エリアの後端アドレスで，初期値はＥ９ＦＦＨ.
ＥＡ５２Ｈ～ＥＡ５３Ｈ	中間言語８１Ｈ～Ｄ５Ｈに対応するテーブルの先頭アドレスで，通常は３３ＢＤＨ.

EA54H　RS－232C用．0の時コミニュケーション・バッファがオーバー・フロー．

EA55H　割り込みレベル．割り込みが無ければFFH．

EA56H　割り込み発生フラグ．通常は0

EA57H　0以外の時カーソルの1段下の行を消去．

EA58H　ターミナル・モード・フラグ．第0ビット←エコー・バックの有無．第1ビット←コントロール・コード表示有無．第2ビット←ASCIIかJISコードか．第3ビット←半二重／全二重．第4ビット←パリティ有／無．第5ビット←パリティ偶／奇．第6ビット←$^{C}{}_{R}$時に$^{L}{}_{F}$も行なうか．第7ビット←ターミナル・モードか否か．

EA59H　カーソルの状態．0で消去，FFHで表示．

EA5AH　ヌル・キャラクタ・コード．初期値は0．

EA5BH　アトリビュート・コード．初期値は0．

EA5CH　ファンクション・キーの表示方法で，50Hの時80文字反転．

EA5DH　スクロール範囲－1．

EA5EH　スクロール開始行＋1．

EA5FH　カーソル・スイッチ．0の時カーソル消去．

EA60H　ファンクション・キー表示スイッチ．0で消去，FFHで表示．

EA61H　カラー／白黒スイッチ．0で白黒モード，FFHでカラー・モ

ード.

ＥＡ６２Ｈ　ＣＲＴ画面の行数.

ＥＡ６３Ｈ　カーソルの縦位置＋１.

ＥＡ６４Ｈ　カーソルの横位置＋１.

ＥＡ６５Ｈ　ＣＲＴ画面の桁数.

ＥＡ６６Ｈ　出力ポート３０Ｈに出力した値.

ＥＡ６７Ｈ　出力ポート４０Ｈに出力した値.

ＥＡ６８Ｈ　ファンクション・キー・フラグ.０以外でファンクション・キーが押された.

ＥＡ６９Ｈ　直前に入力されたキーのキャラクタ・コード.

ＥＡ６ＡＨ　特殊キー入力判定用.０で入力なし，１でシフト，２でＣＴＲＬ，３でカナ，４でカナシフト，５でＧＲＰＨの各キーが押されたことを示す.

ＥＡ６ＢＨ　オート・リピート用で６か４０Ｈ

ＥＡ６ＣＨ～ＥＡ７５Ｈ　入力ポート０～９からの入力データ.

ＥＡ７６Ｈ～ＥＡ７ＢＨ　直前にタイマＩＣから入力した，秒，分，時，日，月，年がＢＣＤコードで入る.Ｎ－ＢＡＳＩＣのＤＡＴＥ＄，ＴＩＭＥ＄で使用.

ＥＡ７ＣＨ～ＥＢ１ＢＨ　ファンクション・キーの内容.

ＥＢ１ＣＨ～ＥＢ１ＦＨ　Ｎ－ＢＡＳＩＣのＰＳＥＴ，ＰＲＥＳＥＴ，ＧＥＴ，ＰＵＴで使用するサブルーチンで，条件によって変化する．

ＥＢ２０Ｈ～ＥＢ２１Ｈ　Ｎ－ＢＡＳＩＣのＬＩＮＥで使用するサブルーチン

ＥＢ２２Ｈ～ＥＢ２３Ｈ　Ｎ－ＢＡＳＩＣのＬＩＮＥで使用するサブルーチン．

ＥＢ２４Ｈ～ＥＢ２６Ｈ　Ｎ－ＢＡＳＩＣのＰＳＥＴ，ＰＲＥＳＥＴで使用するルーチンで，ＲＯＭ内への分岐に用いる．

ＥＢ２７Ｈ～ＥＢ２９Ｈ　Ｎ－ＢＡＳＩＣのＬＩＮＥで使用するルーチンで，ＲＯＭ内への分岐に用いる．

ＥＢ２ＡＨ　ＲＳ－２３２Ｃ，ｃｈ１用．８２５１のコマンド・バイト．

ＥＢ２ＢＨ　ＲＳ－２３２Ｃ，ｃｈ２用．８２５１のコマンド・バイト．

ＥＢ２ＣＨ　未使用．

ＥＢ２ＤＨ　ＲＳ－２３２Ｃの入力ポートのアドレス．ｃｈ１でＣ１Ｈ，ｃｈ２でＣ３Ｈ．

ＥＢ２ＥＨ　ＲＳ－２３２Ｃ，ｃｈ１のポインタ．

ＥＢ２ＦＨ　ＲＳ－２３２Ｃ，ｃｈ２のポインタ．

ＥＢ３０Ｈ　未使用．

ＥＢ３１Ｈ～ＥＢ３２Ｈ　ＲＳ－２３２Ｃ　入力可能バイト数．

ＥＢ３３Ｈ～ＥＢ３９Ｈ　ＲＳ－２３２Ｃ　ｃｈ１用のワーク・エリア．

EB3AH　RS－232C用のワーク・エリア.

EB3BH～EB41H　RS－232C　ch2用のワーク・エリア.

EB42H　RS－232C用のワーク・エリア.

EB43H　RS－232C　ch1への出力カウンタ.

EB44H　RS－232C　ch2への出力カウンタ.

EB45H　未使用.

EB46H　エラー・コード. 0でエラー無し.

EB47H　未使用.

EB48H　プリンタ・ヘッドの位置.

EB49H　各周辺機器への出力スイッチ. 0でCRT, 1～7FHでプリンタ, 80H～FFHでCMTへの出力.

EB4AH　CRT画面の横幅.

EB4BH　CRTの改行値で, N－BASICのPRINTで使用.

EB4CH　未使用.

EB4DH　ESCキーが押されたか否かのフラグ.

EB4EH～EB4FH　N－BASICのINPUT$で使用.

EB50H～EA51H　フリー・エリアの終了アドレス.

ＥＢ５２Ｈ～ＥＢ５３Ｈ　Ｎ－ＢＡＳＩＣで現在実行中の行番号で，ダイレクト・モード時にはＦＦＦＦＨが入る．

ＥＢ５４Ｈ～ＥＢ５５Ｈ　Ｎ－ＢＡＳＩＣのプログラム・エリア開始アドレス．

ＥＢ５６Ｈ～ＥＣ５５Ｈ　Ｎ－ＢＡＳＩＣのプログラム入力時に中間言語に変換するためのワーク・エリア．

ＥＣ５６Ｈ～ＥＣ９５Ｈ　バッファ・エリア．

ＥＣ９６Ｈ～ＥＤ９５Ｈ　Ｎ－ＢＡＳＩＣの入力バッファ．

ＥＤ９６Ｈ～ＥＤ９８Ｈ　テンポラリ・スタック・エリア．

ＥＤ９９Ｈ　カーソルの横座標．

ＥＤ９ＡＨ～ＥＤＢ２Ｈ　ＣＲＴ画面の２５行と対応しており，各行のリンク情報が入る．

ＥＤＢ３Ｈ　ＩＮＰＵＴ時に０でバッド・ファイル・データ，ＲＥＡＤ時に０でアウト・オブ・データ．

ＥＤＢ４Ｈ　スクリーン・エディタ用のカーソル縦座標．

ＥＤＢ５Ｈ　スクリーン・エディタ用のカーソル横座標．

ＥＤＢ６Ｈ　スクリーン・エディタのＣＲＴ画面の横幅．

ＥＤＢ７Ｈ～ＥＤＢ８Ｈ　ターミナル・モード用のワーク・エリア．

ＥＤＢ９Ｈ　ディスクのセクタ・ナンバ．

EDBAH　ドット・グラフィック用のアトリビュート・コード.

EDBBH　ドット・グラフィック用の縦座標.

EDBCH　ドット・グラフィック用の横座標.

EDBDH　グラフィック用のファンクション・コード.

EDBEH～EDBFH　ドット位置のVRAM上のアドレス.

EDC0H～EDC1H　ファンクション・キー・ポインタ．現在入力中のアドレスが入る.

EDC2H　キーボード用の入力ポート.

EDC3H　ドット位置のカーソル縦座標.

EDC4H　ドット位置のカーソル横座標.

EDC5H　ディスク入出時のエラー・カウンタ.

EDC6H　指定ドライブ・ナンバ（0～3）.

EDC7H　接続されているドライブ数.

EDC8H　未使用.

EDC9H　ディスク用入出力ポートの最上位.

EDCAH　N－BASICのLINEで画面消去用に使用.

EDCBH　ディスク関係.

ＥＤＣＣＨ～ＥＤＣＤＨ　未使用.

ＥＤＣＥＨ～ＥＥ４ＤＨ　ＲＳ－２３２Ｃ　ｃｈ１用のインプット・バッファ.

ＥＥ４ＥＨ～ＥＥＣＤＨ　ＲＳ－２３２Ｃ　ｃｈ２用のインプット・バッファ.

ＥＥＣＥＨ　ＲＳＴ　１８ＨでＲＳ－２３２Ｃへ出力するデータの退避用.

ＥＥＣＦＨ　未使用.

ＥＥＤ０Ｈ　ＲＳ－２３２Ｃ用のチャンネル・ナンバ（０～１).

ＥＥＤ１Ｈ　未使用.

ＥＥＤ２Ｈ　ＩＥＥＥ（２）の値.

ＥＥＤ３Ｈ　ＩＥＥＥ（３）の値.

ＥＥＤ４Ｈ　不明.

ＥＥＤ５Ｈ　ＩＥＥＥ－４８８のＳＴＡＴＵＳ.

ＥＥＤ６Ｈ　ＩＥＥＥ－４８８関係.

ＥＥＤ７Ｈ　ＩＥＥＥ－４８８関係.

ＥＥＤ８Ｈ　ＩＥＥＥ－４８８関係.

ＥＥＤ９Ｈ　ＩＥＥＥ－４８８関係.

ＥＥＤＡＨ　ＩＥＥＥ－４８８関係.

ＥＥＤＢＨ　ＩＥＥＥ－４８８で０でＤＩＳＡＢＬＥ，１でＥＮＡＢＬＥ．

ＥＥＤＣＨ　ＩＥＥＥ－４８８関係．

ＥＥＤＤＨ　ＩＥＥＥ－４８８関係．

ＥＥＤＥＨ～ＥＥＤＦＨ　ＩＥＥＥ－４８８のＯＮ　ＳＲＱ　ＧＯＳＵＢで指定する行番号．

ＥＥＥ０Ｈ　ＩＥＥＥ－４８８関係．

ＥＥＥ１Ｈ　ＩＥＥＥ－４８８関係．

ＥＥＥ２Ｈ～ＥＥＥ３Ｈ　ＩＥＥＥ－４８８関係．

ＥＥＥ４Ｈ～ＥＥＥ５Ｈ　ＩＥＥＥ－４８８関係．

ＥＥＥ６Ｈ　ＩＥＥＥ（６）の値．

ＥＥＥ７Ｈ　ＩＥＥＥ（４）の値．

ＥＥＥ８Ｈ　ＩＥＥＥ（５）の値．

ＥＥＥ９Ｈ　ＩＥＥＥ－４８８関係．

ＥＥＥＡＨ　ＩＥＥＥ－４８８関係．

ＥＥＥＢＨ　ＩＥＥＥ－４８８関係．

ＥＥＥＣＨ　ＩＥＥＥ－４８８関係．

ＥＥＥＤＨ　ＩＥＥＥ－４８８関係．

ＥＥＥＥＨ　ＩＥＥＥ－４８８のＣＭＤ　ＤＥＬＩＭで指定したデリメタ・コード．

ＥＥＥＦＨ　ＩＥＥＥ－４８８関係．

ＥＥＥＦＨ　ＩＥＥＥ－４８８のＣＭＤ　ＴＩＭＥＯＵＴで設定したタイム．

ＥＥＦ１Ｈ～ＥＥＦ２Ｈ　ＩＥＥＥ－４８８関係．

ＥＥＦ３Ｈ　ＩＥＥＥ－４８８関係．

ＥＥＦ４Ｈ　ＩＥＥＥ－４８８関係．

ＥＥＦ５Ｈ　ＩＥＥＥ－４８８関係．

ＥＥＦ６Ｈ～ＥＦ２ＢＨ　未使用．

ＥＦ２ＣＨ　ＧＥＴ，ＰＵＴの区別．０でＰＵＴ，ＦＦＨでＧＥＴ．

ＥＦ２ＤＨ～ＥＦ２ＥＨ　ＧＥＴ，ＰＵＴで使用．

ＥＦ２ＦＨ～ＥＦ３０Ｈ　ＧＥＴ，ＰＵＴで使用．

ＥＦ３１Ｈ　ＧＥＴ，ＰＵＴで使用．

ＥＦ３２Ｈ～ＥＦ３３Ｈ　ＧＥＴ，ＰＵＴで使用．

ＥＦ３４Ｈ　ＧＥＴ＠Ａ，ＰＵＴ＠Ａの区別．０でＰＵＴ＠Ａ，ＦＦＨでＧＥＴ＠Ａ．

ＥＦ３５Ｈ　ＧＥＴ，ＰＵＴで使用．

ＥＦ３６Ｈ～ＥＦ３ＢＨ　ＣＭＴでロード，セーブ中のファイル・ネーム．

ＥＦ３ＣＨ　ＣＭＴのエラー発生時に，リード・エラーとするかどうかこフラグで，０の時リード・エラーとする．

ＥＦ３ＤＨ　未使用．

ＥＦ３ＥＨ～ＥＦ４３Ｈ　ＣＭＴで現在ロード中のファイル・ネーム．

ＥＦ４４Ｈ　配列変数処理に使用．

ＥＦ４５Ｈ　フローティング・アキュームレータの型を示す数字．２で整数型，３で文字型，４で単精度型，８で倍精度型．

ＥＦ４６Ｈ　ＬＩＳＴ編集に使用．

ＥＦ４７Ｈ　ＬＩＳＴ編集に使用．

ＥＦ４８Ｈ～ＥＦ４９Ｈ　実行中のプログラムのテキスト上のアドレス．

ＥＦ４ＡＨ　実行中の命令に付随するコントロール・コード．ＢＨ～ＦＨ，１１Ｈ～１ＤＨ，１ＦＨ．

ＥＦ４ＢＨ　数値定数の型．２，４，８．

ＥＦ４ＣＨ～ＥＦ５３Ｈ　プログラム上の数値定数を１度ここに移してからフローティング・アキュームレータに格納する．

ＥＦ５４Ｈ～ＥＦ５５Ｈ　Ｎ－ＢＡＳＩＣプログラム・エリアの終了アドレス．

ＥＦ５６Ｈ～ＥＦ５７Ｈ　ストリング・ディスクリプタの格納アドレス．

ＥＦ５８Ｈ～ＥＦ７８Ｈ	１１組のストリング・ディスクリプタ.
ＥＦ７９Ｈ～ＥＦ７ＡＨ	フリー・エリアの終了アドレス.
ＥＦ７ＢＨ～ＥＦ７ＣＨ	ＤＩＭ，ＦＮ，ＮＥＸＴで使用.
ＥＦ７ＤＨ～ＥＦ７ＥＨ	文字型配列変数の実際のアドレス.
ＥＦ７ＦＨ～ＥＦ８０Ｈ	ＦＯＲでテキスト・ポインタ退避用.
ＥＦ８１Ｈ～ＥＦ８２Ｈ	現在読み込んでいるデータの行番号.
ＥＦ８３Ｈ	ＦＯＲ，ＥＲＡＳＥ，ＤＩＭ，ＦＮで使用.
ＥＦ８４Ｈ	ＩＮＰＵＴ，ＲＥＡＤ，ＰＲＩＮＴＵＳＩＮＧで使用.
ＥＦ８５Ｈ～ＥＦ８６Ｈ	ＥＲＡＳＥ，ＦＯＲ，ＬＥＴ，ＮＥＸＴで使用.
ＥＦ８７Ｈ	ＧＯＴＯで使用.
ＥＦ８８Ｈ	ＡＵＴＯ実行中フラグ.
ＥＦ８９Ｈ～ＥＦ８ＡＨ	ＡＵＴＯカウンタ.
ＥＦ８ＢＨ～ＥＦ８ＣＨ	ＡＵＴＯ増分.
ＥＦ８ＤＨ～ＥＦ８ＥＨ	プログラム・テキストの先頭アドレス.
ＥＦ８ＦＨ～ＥＦ９０Ｈ	ＦＯＲ，ＮＥＸＴ，ＲＥＴＵＲＮで使用.
ＥＦ９１Ｈ～ＥＦ９２Ｈ	システム変数ＥＲＬ.

ＥＦ93Ｈ～ＥＦ９４Ｈ　直前に処理した行番号で，ＬＩＳＴ．等の「．」で使用．

ＥＦ９５Ｈ～ＥＦ９６Ｈ　ＲＥＳＵＭＥで使用．

ＥＦ９７Ｈ～ＥＦ９８Ｈ　エラー関係．

ＥＦ９９Ｈ　エラー発生フラグ．

ＥＦ９ＡＨ～ＥＦ９ＢＨ　ＤＩＭ，ＦＮ，ＮＥＸＴで使用．

ＥＦ９ＣＨ～ＥＦ９ＤＨ　ブレークした行番号．

ＥＦ９ＥＨ～ＥＦ９ＦＨ　ＣＯＮＴを行なうための中断アドレス．０でＣＯＮＴ不可能．

ＥＦＡ０Ｈ～ＥＦＡ１Ｈ　変数エリアの開始アドレス．

ＥＦＡ２Ｈ～ＥＦＡ３Ｈ　配列変数エリアの開始アドレス．

ＥＦＡ４Ｈ～ＥＦＡ５Ｈ　フリー・エリアの開始アドレス．

ＥＦＡ６Ｈ～ＥＦＡ７Ｈ　現在読み込んでいるデータのアドレス．

ＥＦＡ８Ｈ～ＥＦＣ１Ｈ　ＤＥＦで指定した変数の型で，Ａ～Ｚの２６バイト．

ＥＦＣ２Ｈ～ＥＦＣ３Ｈ　ＦＮで使用．

ＥＦＣ４Ｈ～ＥＦＣ５Ｈ　ＦＮで使用．

ＥＦＣ６Ｈ～Ｆ０２９Ｈ　ＦＮ用引数の型，名前と処理アドレスが入る．

Ｆ０２ＡＨ～Ｆ０９１Ｈ　ＦＮ用引数の型，名前と処理アドレスが入る．

F092H	DIMで使用.
F093H～F094H	DIMで使用.
F095H	FNで使用.
F096H～F097H	ガベージ・コレクション時に使用.
F098H～F099H	FNで使用.
F09AH～F0A1H	SWAP実行時の変数退避用.
F0A2H	トレース実行フラグ. 0でTROFF, FFHでTRON.
F0A3H	フローティング・アキュームレータの無効桁.
F0A4H～F0ABH	フローティング・アキュームレータ.
F0ACH	フローティング・アキュームレータの符号保持用.
F0ADH	サブ・フローティング・アキュームレータの無効桁.
F0AEH～F0B5H	サブ・フローティング・アキュームレータ.
F0B6H～F0D0H	数値出力用バッファ.
F0D1H～F0D8H	倍精度除算用レジスタ.
F0D9H～F0DFH	倍精度乗算用レジスタ.
F0E1H～F16AH	命令のジャンプ・テーブル.

ＦＩ６ＢＨ～Ｆ２１５Ｈ　処理のジャンプ・テーブル.

Ｆ２１６Ｈ～Ｆ２ＦＦＨ　未使用.

Ｆ３００Ｈ～ＦＥＢ７Ｈ　ＶＲＡＭ.

ＦＥＢ８Ｈ～ＦＦ２ＦＨ　ＣＲＴ画面クリア用のデータ１行分１２０バイト.

ＦＦ３０Ｈ～ＦＦ３１Ｈ　モニタのＳコマンドでエラー発生時のアドレス退避用.

ＦＦ３２Ｈ　未使用.

ＦＦ３３Ｈ　モニタのベリファイ・スイッチ.０でロード，０ＤＨでベリファイ.

ＦＦ３４Ｈ～ＦＦ３５Ｈ　モニタに制御が移った時のＨＬレジスタ対退避用.

ＦＦ３６Ｈ～ＦＦ３７Ｈ　モニタに制御が移った時のスタック・ポインタ（ＳＰ）対避用.

ＦＦ３８Ｈ　モニタのＣＲＴ表示バイト数切り換え用.０でＳコマンド，ＦＦＨでＤコマンド.

ＦＦ３９Ｈ～ＦＦ３ＡＨ　モニタのＴＭコマンドで不良メモリのアドレスが入る.

ＦＦ３ＢＨ　ＴＭコマンドで書き込んだデータ.

ＦＦ３ＣＨ　ＴＭコマンドで読み出したデータ.

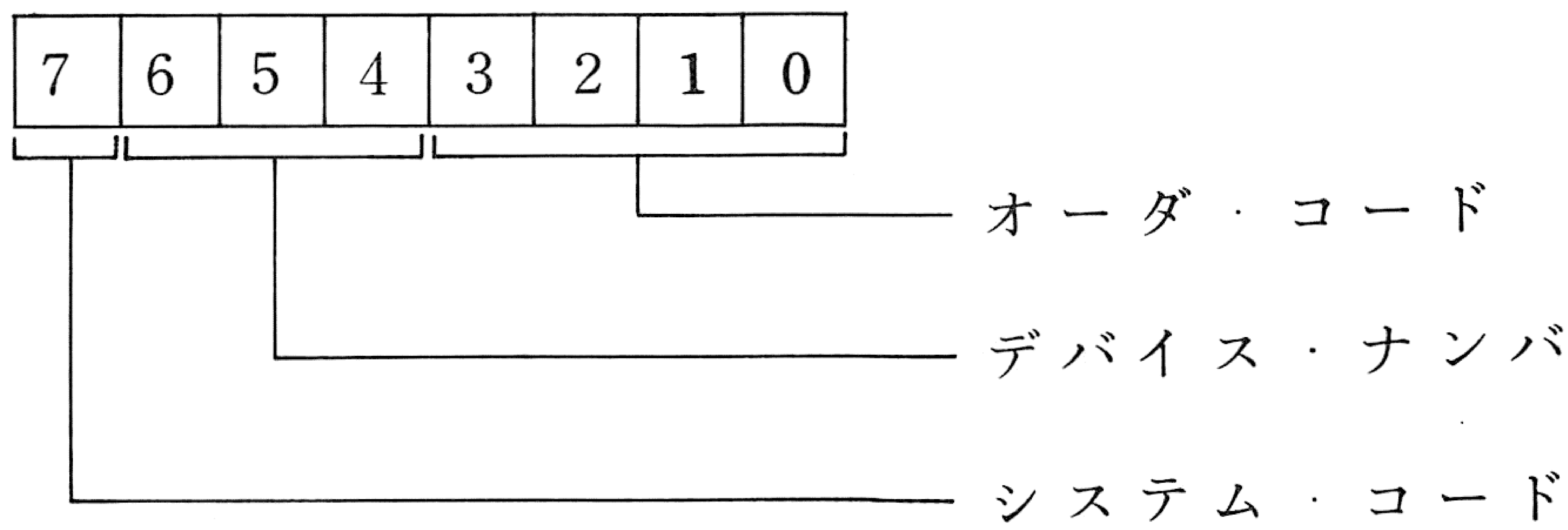
ＦＦ３ＤＨ～ＦＦＦＥＨ　リセット時のスタック・エリア.

ＦＦＦＦＨ　未使用.

2
IOCSの解説

2.1　Ｉ／Ｏアドレスの構成

ＰＣ－８００１では，Ｚ８０に用意されている８ビットのＩ／Ｏアドレスの最上位ビットをシステム・コードとして，第６ビット～第４ビットをデバイス・ナンバの指定に，又下位４ビットをオーダ・コードとして用いています．



システム・コードは，ＰＣ－８００１内部のＩ／Ｏインターフェースを用いるか，外部のものを用いるかの区別に使用されており，前者の場合０，後者の場合は１にセットされます．

つまり，ＰＣ－８００１内部のＩ／Ｏインターフェースを使用している場合には常に０にしておけば良い事になります．

デバイス・ナンバは各Ｉ／Ｏインターフェースの中でどのインターフェースを制御するかを指定するためのもので８種類のインターフェースを選択する事ができます．

ナンバ	内部のＩ／Ｏインターフェース
０００	キーボード
００１	プリンタ／タイマ
０１０	ＣＭＴ／ＳＩＯ（８２５１）
０１１	コントロール１
１００	コントロール２
１０１	ＣＲＴコントローラ（３３０１）
１１０	ＤＭＡコントローラ（８２５７）
１１１	未使用

オーダ・コードはシステム・コードとデバイス・ナンバによって指定されたインターフェースに関して実際の働きを指定するものです．

2.2 主要Ｉ／Ｏポート・マッピング

００Ｈ～０９Ｈ	キーボード入力.
１０Ｈ	プリンタ・データ出力.
２０Ｈ	８２５１へのデータ・ワード出力.
２１Ｈ	８２５１へのコントロール・ワード出力.
３０Ｈ	コントロール・ポート１
４０Ｈ	コントロール・ポート２
５０Ｈ	３３０１（ＣＲＴコントローラ）へのデータ出力.
５１Ｈ	３３０１（ＣＲＴコントローラ）へのコントロール出力.
６４Ｈ	８２５７（ＤＭＡコントローラ）へのＣＨ－２アドレス・セット出力.
６５Ｈ	８２５７（ＤＭＡコントローラ）へのＣＨ－２ターミナル・カウンタ・セット出力.
６８Ｈ	８２５７（ＤＭＡコントローラ）へのモード・セット出力およびステータス入力.

ＦＣＨ	ディスク・ユニットからの８ビット・データ入力.
ＦＤＨ	ディスク・ユニットへの８ビット・データ出力.
ＦＥＨ	上位４ビットが制御信号入力，下位４ビットが制御信号出力.
ＦＦＨ	ディスク・ユニット用の８２５５（ＰＩＯ）モード・セット出力.

2.3　キーボード

ＰＣ－８００１のキーボードは，エンコーダ等が用いられておらず，パーソナル・コンピュータでは最も一般的な，ソフトウェア・キー・スキャンによって押されたキーの判別を行なっています．

キーボードのそれぞれのキーは，次頁に示すようなマトリクスで構成されており，おのおのの列がデータ・バスに直接接続されていますので，入力ポートからの入力によって簡単に，押されたキーを判別する事ができます．

実際には，Ｎ－ＢＡＳＩＣのＩＮＰやＺ８０のＩＮによって，データ・バスの値を入力し，キーが押されていれば，その列のデータ・バスに対応したビットが0として入力されます．

実例をいくつかあげますと，たとえば，Ｋのキーが押されている場合に

　　ＩＮ　　Ａ，（０３Ｈ）

を行なえば，Ａレジスタに2進数で，

　　１１１１　０１１１

が入力され，ＫとＯが同時に押されている場合に

　　ＩＮ　　Ａ，（０３Ｈ）

を行なえば，

　　０１１１　０１１１

が入力される事になります．

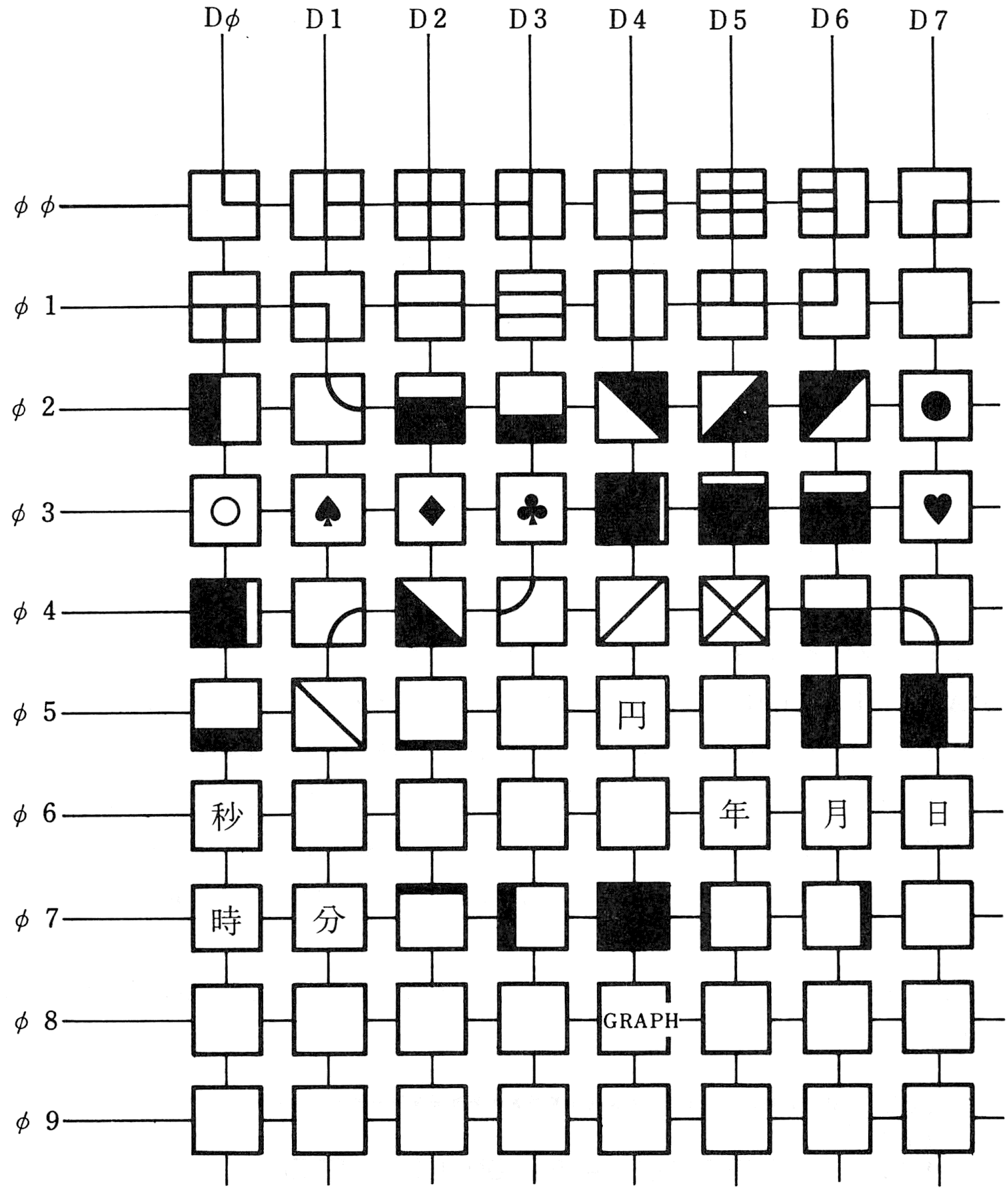
ポートから入力されたデータを，キャラクタ・コードに変換する事が必要な場合も多いと思われますが，Ｎ－ＢＡＳＩＣでは，１０Ｅ５Ｈ～１１７ＣＨ番地の変換テーブルを利用して，変換を行なっています．

2.3.1 キーボード・マトリクス1

(データ・バス)

(インプットアドレス)	Dφ	D1	D2	D3	D4	D5	D6	D7
φφ	0	1	2	3	4	5	6	7
φ1	8	9	*	+	=	,	.	RETURN
φ2	@ 〃	A チ	B コ	C ソ	D シ	E イ	F ハ	G キ
φ3	H ク	I ニ	J マ	K ノ	L リ	M モ	N ミ	O ラ
φ4	P セ	Q タ	R ス	S ト	T カ	U ナ	V ヒ	W テ
φ5	X サ	Y ン	Z ツ ッ	[。「	¥ _	] ム 」	^ へ	− = ホ
φ6	0 ワ ヲ	1 ! ヌ	2 " フ	3 # ア ァ	4 $ ウ ゥ	5 % エ ェ	6 & オ ォ	7 ' ヤ ャ
φ7	8 (ユ ュ	9) ヨ ョ	: * ケ	; + レ	, < ネ 、	. > ル 。	/ ? メ ・	− ロ
φ8	HOME CLR	↓ ↑	← →	INS DEL	GRAPH	カナ	SHIFT	CTRL
φ9	STOP	f・1	f・2	f・3	f・4	f・5	SPACE	ESC

2.3.2 キーボード・マトリクス 2



2.3.3 ファンクション・キーについて

N－BASICでは，[f・1]から[f・10]まで，計10個のプログラマブル・ファンクション・キーを使用する事が可能ですが，それぞれのキーの内容は，EA7CH～EB1BHにキャラクタ・コードで収納されており，N－BASICのコールド・スタート時に，ROM上の3A81H～3B20H番地の値が，そのまま転送されてきます．

すなわち，ファンクション・キー全ての内容を，コールド・スタート時の初期状態に再設定するには次の様なサブルーチンを実行すれば良い事になります．

```
LD      HL，3A81H
LD      DE，EA7CH
LD      BC，00A0H
LDIR
RET
```

2.3.3.1 ファンクション・キー・バッファ

RAM上のEA7CH番地から始まるファンクション・キー・バッファは，1つのキーに対して16バイト分が用意してありますが，定義内容の最後には必ず0を入れておかなければなりませんから，実質的には15文字までしか登録できない事になります．しかし，N－BASICのPOKEやマシン語等によって，定義内容の区切りである0の入っているアドレスの値を書き換えればバッファの許す範囲内で文字列をつめ込む事が可能です．

次に，[f・1]キーにアルファベット大文字のA～Zまでの26文字を定義するN－BASICのプログラム例を挙げます．

```
100 '********************************
110 '
120 '   STORE ALPHABET ON F1 BUFFER
130 '
140 '********************************
150 '
160 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"+CHR$(0)
170 FOR I=1 TO LEN(A$)
180   POKE &HEA7C+I-1,ASC(MID$(A$,I,1))
190 NEXT I
200 END
```

このプログラム実行後に，[f・1]キーを押せばA～Zまでの２６文字をキー入力したのと同じ状態になり，[f・2]キーを押した場合は，Q～Zの入力と同じです．また，この様なファンクション・キーの使い方では，定義内容がバッファの最終アドレスである，ＥＢ１ＢＨ番地以上にあふれない事と最後には必ず0を入れる事に注意してください．

つまり，[f・1]キーには最大で，１６×１０－１＝１５９文字，[f・１０]キーには１５文字しか定義できない事になります．

2.3.3.2　ファンクション・キー・フラグとファンクション・キー・ポインタ

ファンクション・キー・バッファの他に重要なものとしては，ＥＡ６８Ｈ番地のファンクション・キー・フラグとＥＤＣ０Ｈ番地から2バイトに渡るファンクション・キー・ポインタが有ります．

ファンクション・キー・フラグは，ファンクション・キーが押されているか否かのフラグとして用いられており通常は0が入っていますが，ファンクション・キーが押されている場合には1が入ります．

キー・ポインタは，ファンクション・キー・バッファ内のどの文字を入力キーとして受け付けて行くかが，２バイトのアドレスとして下位8ビット，上位8ビットの順に入っています．

つまり，この3バイトのワーク・エリアの内容を強制的に書き換えてしまう事によってＮ－ＢＡＳＩＣやモニタ等に，任意のキーが押されたごとく思い込ませる事も可能です．もちろん，キー・バッファを従来のバッファ・エリアで無く好きなアドレスに設ける事ができますから，極端に言えば，空いているＲＡＭエリア全てをバッファとして用いて2万文字以上を定義する事も不可能ではありません．

2.3.3.3 ファンクション・キー関係のワーク・エリア

▷ファンクション・キー・フラグ

EA68H

▷ファンクション・キー・バッファ

f	スタート アドレス	エンド アドレス	エンド マーク	初期内容
1	EA7CH	EA8AH	EA8BH	$^{H}{}_{T}$
2	EA8CH	EA9AH	EA9BH	auto
3	EA9CH	EAAAH	EAABH	go to
4	EAACH	EABAH	EABBH	list
5	EABCH	EACAH	EACBH	run $^{C}{}_{R}$
6	EACCH	EADAH	EADBH	time$
7	EADCH	EAEAH	EAEBH	key
8	EAECH	EAFAH	EAFBH	print
9	EAFCH	EB0AH	EB0BH	list. $^{C}{}_{R}$ ↑ ↑
10	EB0CH	EB1AH	EB1BH	cont $^{C}{}_{R}$

▷ファンクション・キー・ポインタ

EDC0H
EDC1H

2.4 プリンタ

プリンタへデータを出力するのであれば，N－BASIC内のシステム・サブルーチンを利用するだけで，十分に用が足ります．

1バイトだけを出力するのであれば，Aレジスタにデータを入れてコールする，0D60H番地からのプリンタ出力サブルーチンが最も手軽に利用できますし，リスタート命令でもコールする事ができる40A6H番地の各周辺機器への1バイト出力なども便利です．

2バイト以上に渡るデータや，文字列などを出力する場合には，通常はCRT画面への文字列表示用に使われている52EDH番地の文字列出力サブルーチンを，EB49H番地の出力スイッチを切り換えてプリンタへの出力に使用する事ができます．

他にもCRT画面への表示用に使われている多くのサブルーチンのほとんどが出力スイッチの切り換えのみでプリンタに出力可能です．いろいろ試してみると良いでしょう．

また，直接I／Oポートをドライブして，プリンタへの出力を行なう場合には，コントロール・ポートに，BUSYおよび$\overline{\mathrm{ACK}}$信号が入力しており，出力ポート10H番地をプリンタへの8ビット出力データの出力用に使用します．

2.5 タイマ

2.5.1 タイマ処理ルーチンのワーク・エリア

N－BASIC内のタイマ制御ルーチンではワーク・エリアとして，EA76H～EA7BH番地までを使用しています．ワーク・エリアは，タイマ関係のコマンドが用いられるたびに更新され，以下の内容が全てBCDコードにて格納されています．

EA76H番地 ⟵ SECOND（秒）
EA77H番地 ⟵ MINUTE（分）
EA78H番地 ⟵ HOUR　（時）
EA79H番地 ⟵ DAY　（日）
EA7AH番地 ⟵ MONTH（月）
EA7BH番地 ⟵ YEAR　（年）

次に，N－BASICのシステム・サブルーチンを利用して，タイマICを制御する方法を項目別に分けて説明して行きたいと思います．

2.5.2 タイマの初期設定

N－BASICの電源投入時にタイマICは，

TIME＄ ← 00：00：00
DATE＄ ← 79／01／01

に，セットされます．同じ事を，164CH番地からのサブルーチンをコールする事によって行なう事ができます．

EA76H～EA7BH番地の内容も同時に初期設定が行なわれます．

この，164CH番地のサブルーチンは，N－BASICのコールド・スタートおよびホット・スタート・ルーチンで最初にコールされています．

2.5.3 タイマ・リード

タイマＩＣの値を読み込むためには，１６０２Ｈ番地からのサブルーチンを利用します．

読み込んだ値は，ＥＡ７６Ｈ～ＥＡ７ＢＨ番地のワーク・エリアに，ＢＣＤコードで格納されます．

2.5.4 タイマ・ライト

逆に，ワーク・エリアのＢＣＤコードをタイマＩＣに書き込む役割りを持っているのが１６６３Ｈ番地からのサブルーチンです．

ただし，いずれの場合にも，ＹＥＡＲ（年）の値はタイマＩＣとのやり取りを行なっていません．ですから，Ｎ－ＢＡＳＩＣのＤＡＴＥ＄は再設定を行なわない限りは常に７９年のままで，また，１２月３１日から１月１日に変化してもＹＥＡＲ（年）の値は変化しません．

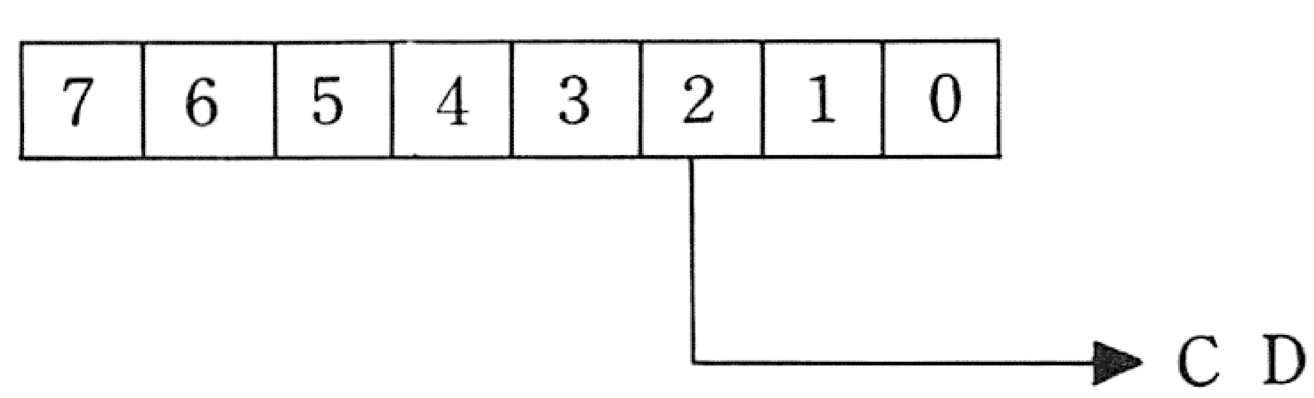
2.6　ＣＭＴおよびＳＩＯ

2.6.1　シリアル・インターフェイスの関連ポート

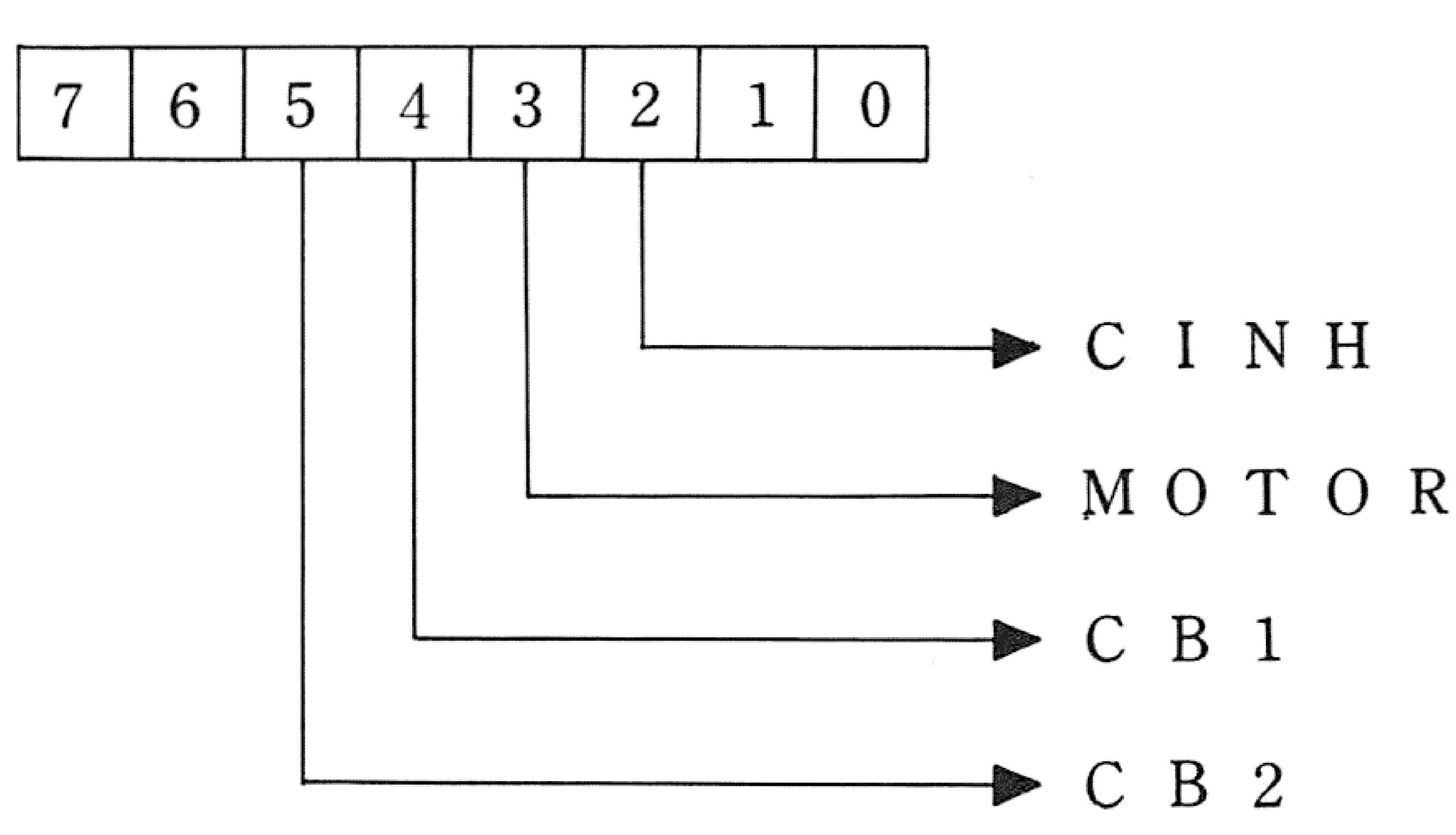
Ｎ―ＢＡＳＩＣでは，シリアル・インターフェイスのコントロール用として，プログラマブル・コミュニケーション・インターフェイス，８２５１を使用しています．

また，ＣＭＴ／ＳＩＯのインターフェイスをコントロールするために関連のあるポートとしては，出力ポート３０Ｈ番地の，第５ビットから第２ビットまでと，入力ポート４０Ｈ番地の第２ビットをあげる事ができます．

入力ポート４０Ｈ番地



出力ポート３０Ｈ番地



2.6.1.1　ＣＩＮＨ

ＣＭＴには，実際に録音するデータの前後に，１２００Ｈｚのキャリア信号を書き込みますが，このキャリアを書くか書かないかの決定をこのビットによって行ないます．

キャリアを書いていない場合には，データが０の時に１２００Ｈｚの，データが１の時には２４００Ｈｚの信号を録音します．

ＣＩＮＨ	信　　号
0	データに対応した信号
1	１２００Ｈｚのキャリア信号

2.6.1.2　MOTOR

ＣＭＴをリモート・コントロールするために，モータ・リレーの状態を制御するためのビットです．

ＭＯＴＯＲ	状　　態
0	モータ・リレー・オフ
1	モータ・リレー・オン

2.6.1.3　CD

この入力ポートは，ＣＭＴインターフェースとＳＩＯインターフェースの場合で利用方法が異なり，ＣＭＴの場合にはキャリア信号を読み込んで検出したとき１にセットされ，ＳＩＯの場合には，ＳＩＯコネクタのＣＤ端子の状態が０か１かを調べる事ができます．

2.6.1.4　CB1，CB2

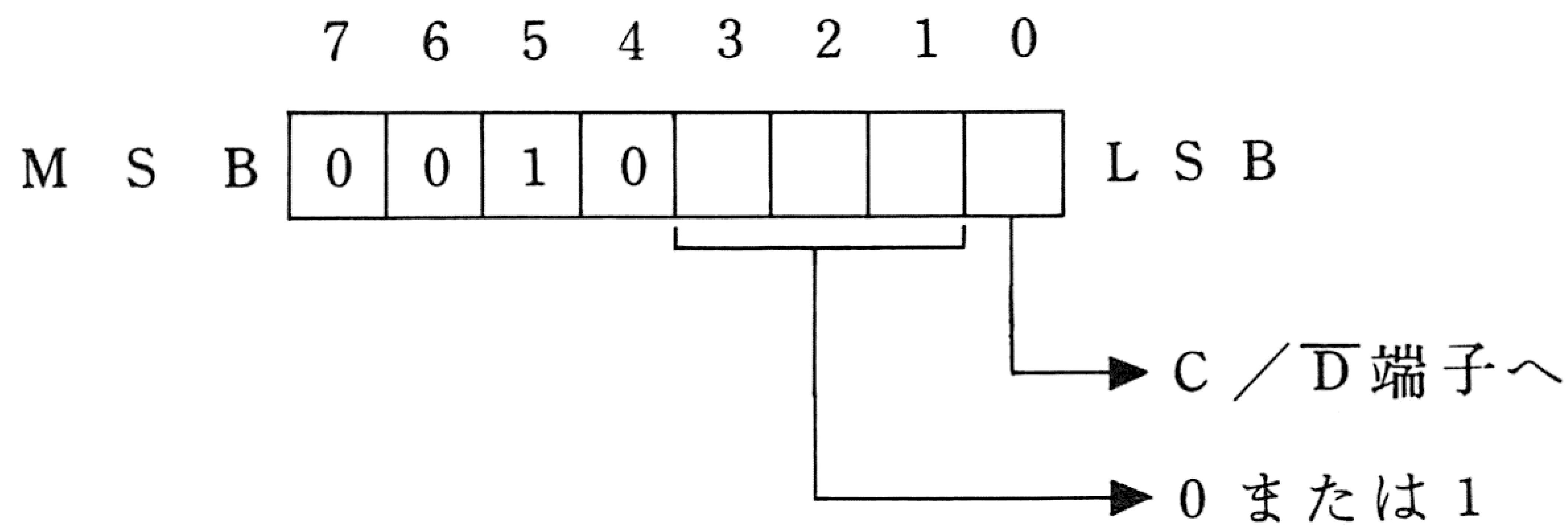
８２５１を，ＣＭＴインターフェースに使用するのか，ＳＩＯインターフェースに使用するのかを切り換えるためと，ＣＭＴインターフェースとして使用する場合の転送速度を選択するために使用するものです．

ＣＢ１，ＣＢ２の組み合わせは次のようになっており，ＣＢ２によって，ＣＭＴとＳＩＯを切り換えます．

ＣＢ２	ＣＢ１	機　　能
0	0	ＣＭＴ　６００ボー
0	1	ＣＭＴ　３００ボー
1	0	ＳＩＯ
1	1	使用禁止

2.6.2　８２５１のＩ／Ｏアドレス

Ｎ－ＢＡＳＩＣでは，８２５１のＩ／Ｏアドレスを次のように構成しており，アドレス・バスの最下位ビットが，８２５１のＣ／$\overline{\text{D}}$端子に接続されています．



８２５１のＣ／$\overline{\text{Ｄ}}$端子は，データ・バスの内容がデータの入出力か，ステータスのリードまたは８２５１の制御なのかを決定するためのもので，０の場合は前者で１の場合には後者になります．

以上の事から，８２５１のＩ／Ｏアドレスは，２０Ｈ番地と２１Ｈ番地を用いれば良い事がわかります．

２０Ｈ番地：データの入出力

２１Ｈ番地：ステータスのリード及び８２５１の制御

2.6.3　８２５１の使用方法

８２５１は，データ通信用に開発されたＵＳＡＲＴで，プログラムによって同期／非同期通信の選択，送受信フォーマットの設定，モデム等の制御，ステータスによるエラー・チェックなどを行なう事ができます．

ＵＳＡＲＴとは

Universal
Synchronous
Asynchronous
Resiver
Transmitter

の頭文字を取ったものです．

2.6.3.1　８２５１の初期設定

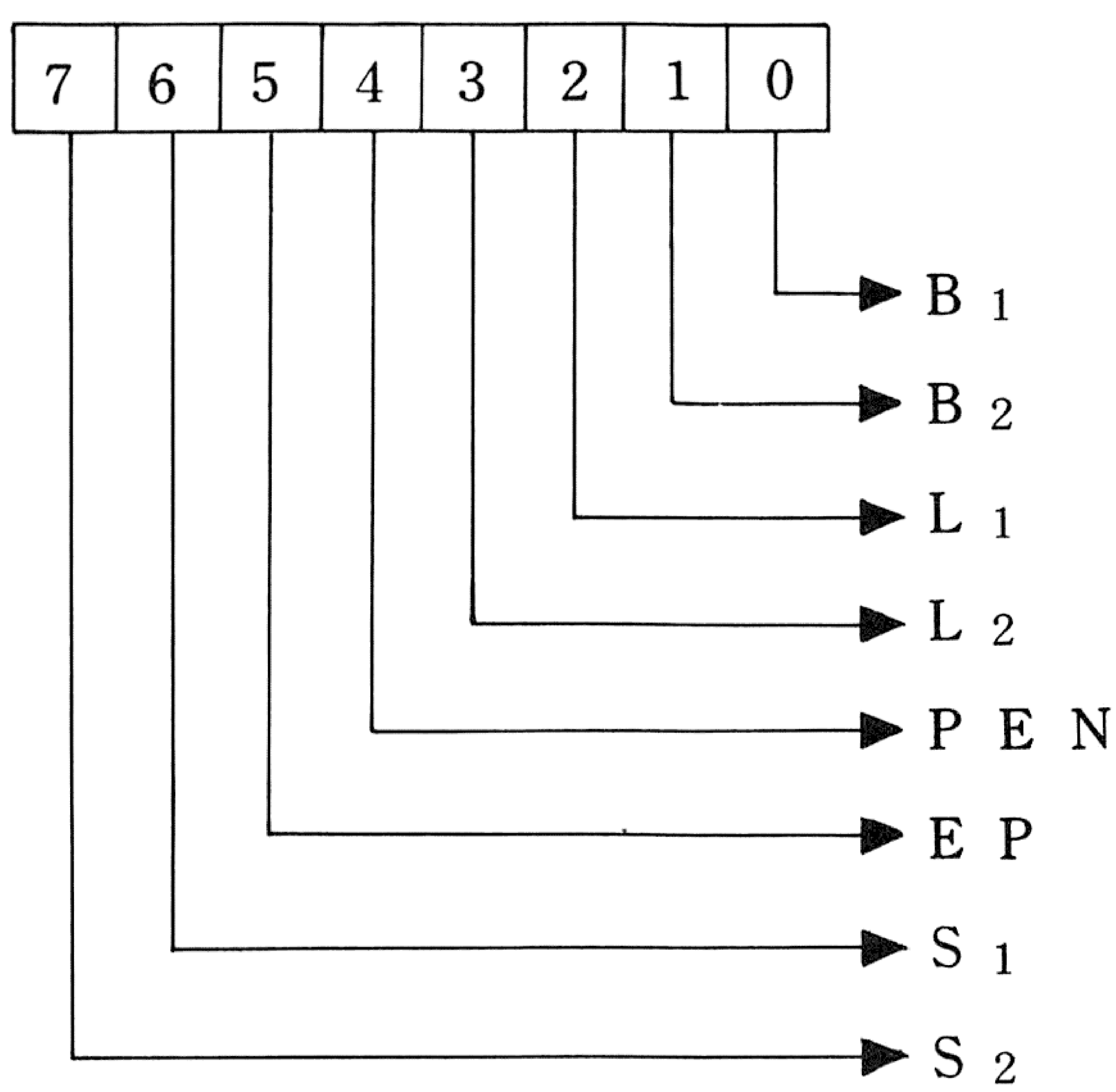
まず最初に，８２５１の初期設定を行なわなければなりません．

モード設定とコマンド設定の２種類がありますが，８２５１では，ハードまたはコマンドによってリセットされた直後に１回だけモード設定を行なう事ができ，以後２１Ｈ番地に出力したものは全てコマンドとして受け取られます．

ところで，ユーザーのプログラム中から，８２５１を制御する場合には，８２５１がリセット直後なのか，すでにモード設定が行なわれたのかを知る事ができません．そこで，ユーザーのプログラム中で８２５１をリセットする場合には１度ダミーのデータを送って完全にコマンド待ちにしてから，リセット・コマンドを送らなければなりません．

2.6.3.2　８２５１のモード設定

モード設定のために，８２５１に与えるデータの各ビットの意味を次に示します．



2.6.3.2.1　S_1，S_2

ストップ・ビットの長さを指定するために使用します．S_1，S_2とストップ・ビットの長さは次のような関係になります．

S_2	S_1	ストップ・ビット長
0	0	ストップ・ビット無効
0	1	１ビット分
1	0	1.5ビット分
1	1	２ビット分

2.6.3.2.2　EP

パリティ・ビットをつける場合に，偶数か奇数かの指定を行ないます．

1にセットした時は偶数パリティとなり，0の場合には奇数パリティが指定されます．

2.6.3.2.3　PEN

パリティ・ビットを付けるか否かの指定に使い，1の時イネーブル，0の時ディスエイブルとなります．

2.6.3.2.4　L_1，L_2

転送するデータのキャラクタ長を指定するために使用します．N－BASICの場合にはJIS8単位を使用していますから，8ビット必要です．

L_2	L_1	キャラクタ長
0	0	5ビット
0	1	6ビット
1	0	7ビット
1	1	8ビット

2.6.3.2.5　B_1，B_2

両者共に0を指定した場合は同期通信モードで，それ以外は非同期通信モードになります．

一般には非同期通信を使用し，その場合には転送速度の指定も同時に行ないます．

B_2	B_1	通信モード(転送速度)
0	0	同期通信モード
0	1	非同期通信モード(×01)
1	0	非同期通信モード(×16)
1	1	非同期通信モード(×64)

転送速度は，8251に入力されているクロックの周波数と次の様な関係になります．

$$\text{ボーレート} = \frac{\text{クロックの周波数}}{01 \text{ or } 16 \text{ or } 64}$$

たとえば，クロックが4.8KHzの時に，×16を指定すれば300ボーになります．

2.6.3.2.6　モード設定例

８２５１に4．８KHzのクロックが与えられている場合に次の条件を満たすモードに設定する場合は，８２５１がリセットされた直後に，データCEHを２１H番地の出力ポートに出力する事になります．

▷非同期通信モードで３００ボー

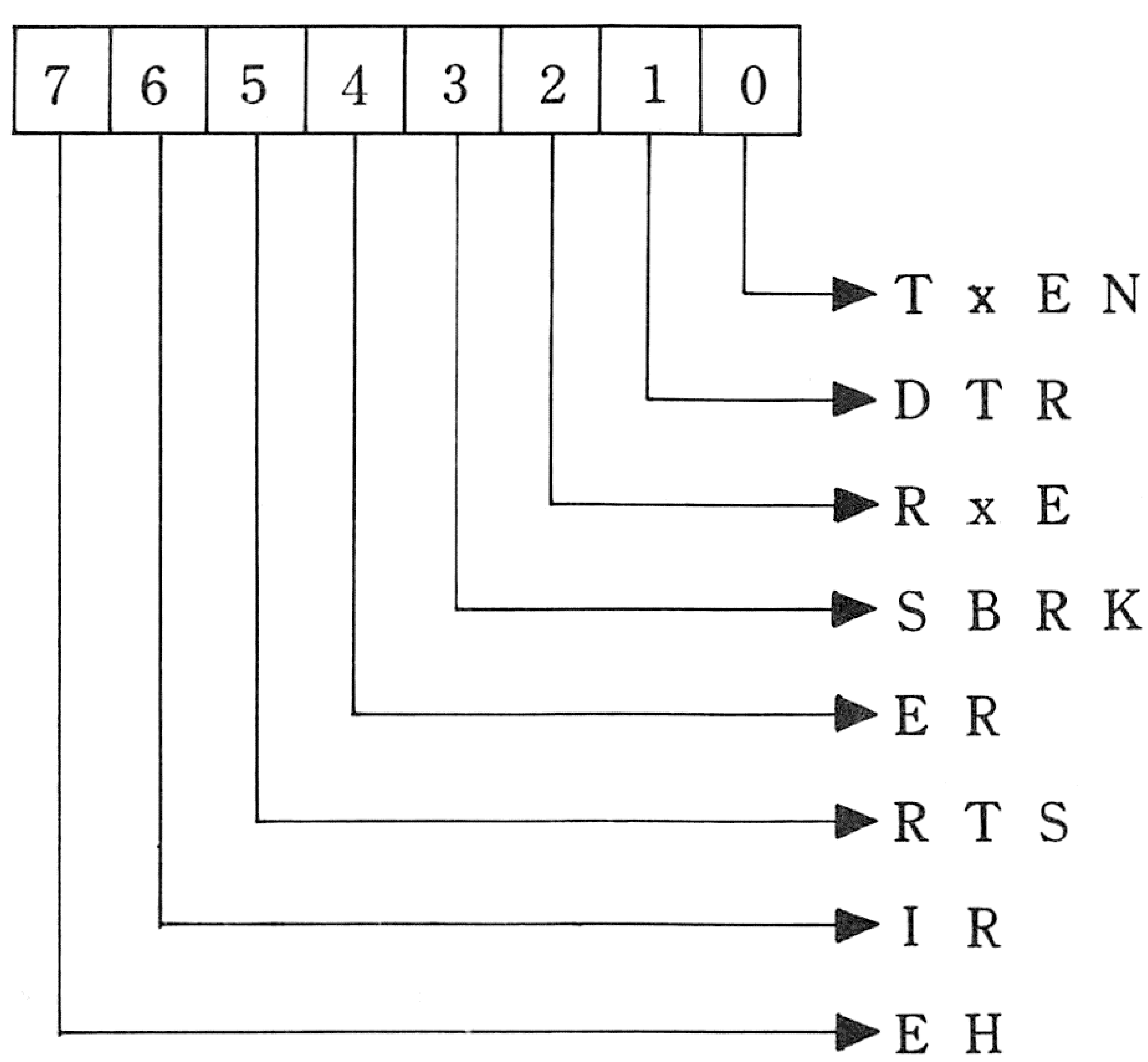
▷パリティ・ビット無効

▷８ビットのキャラクタ長

▷２ビットのストップ・ビット

2.6.3.3　８２５１のコマンド設定

次に，コマンド設定のために，８２５１に与えるデータの各ビットの意味を示します．



2.6.3.3.1　EH

同期通信の場合に使用するビットで，HUNTモードで１にセットされている場合にはSYNCキャラクタの検出を始めさせます．

2.6.3.3.2　IR

８２５１を，ソフト・ウェアからリセットするためのビットで，１にセットされた場合には，８２５１がイニシアル状態となり，以後１回だけはモード設定を受け付けます．

2.6.3.3.3　RTS

センド要求を行なうためのビットで，$\overline{\text{RTS}}$端子をコントロールする事ができ，このビットを１にセットすると$\overline{\text{RTS}}$出力が０にリセットされます．

2.6.3.3.4　ER

ステータスのエラー・フラグ全てをリセットするためのもので，ＰＥ，ＯＥ，ＦＥをリセットします．

2.6.3.3.5　SBRK

センド・ブレーク・キャラクタと言い，通常は０にして使用しますが，ＴｘＤの送信データを０にする場合のみセットします．

2.6.3.3.6　RxE

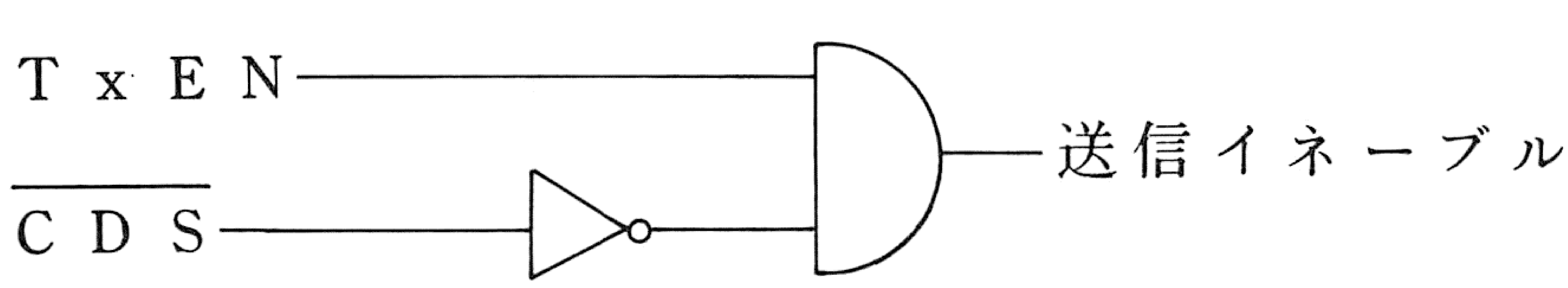
受信イネーブルの指定に使用し，１にセットした場合はイネーブル，０の場合にはディスエイブルとなります．

2.6.3.3.7　DTR

データ・ターミナル・レディと言い，$\overline{\text{DTR}}$端子をコントロールするためのビットです．

2.6.3.3.8　TxEN

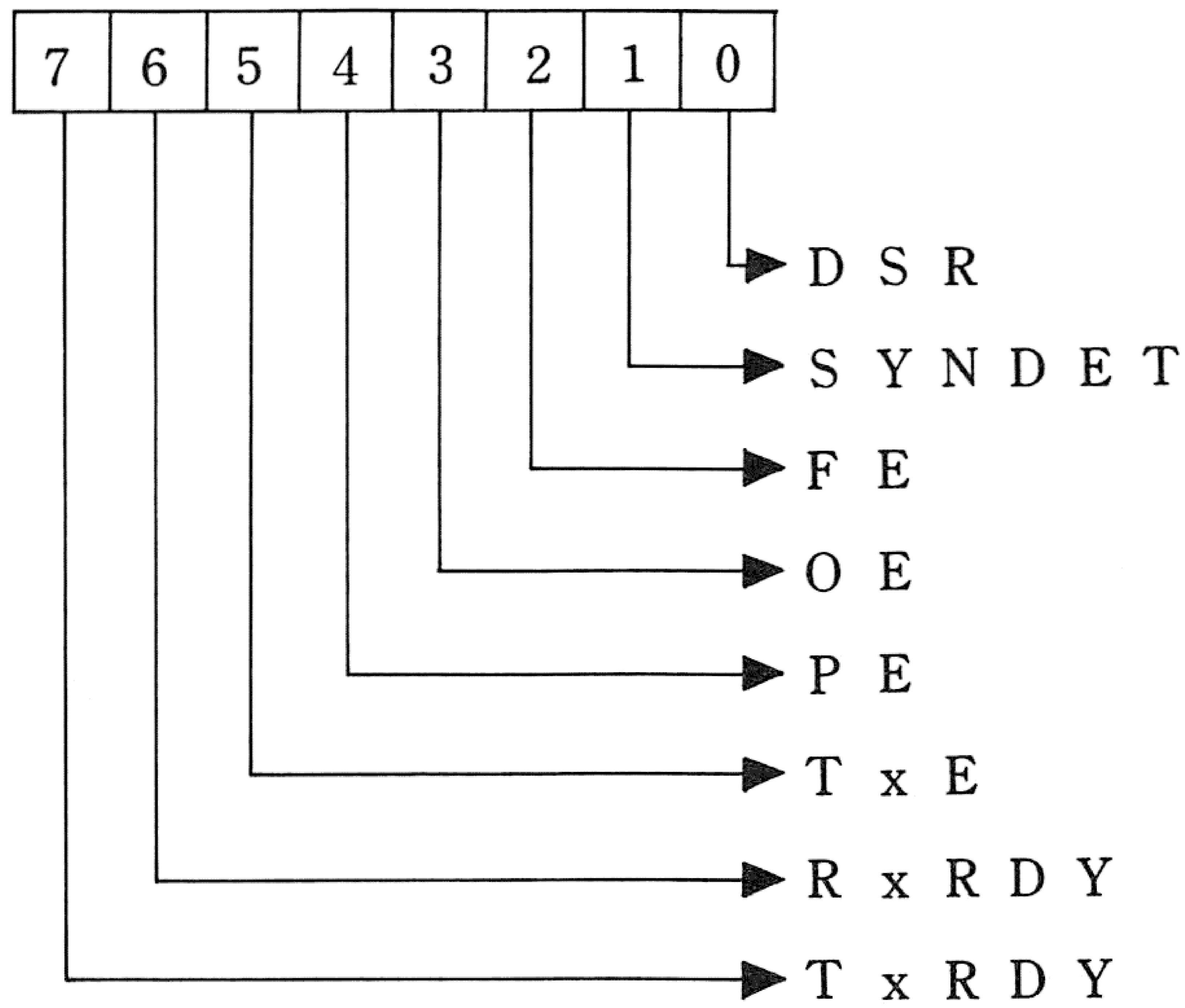
送信イネーブルの指定に使用し，８２５１の$\overline{\text{CDS}}$端子と次の関係があります．



つまり，ＴｘＥＮのビットをイネーブルにセットしておけば$\overline{\text{CDS}}$端子が０になる事で送信可能になり，$\overline{\text{CDS}}$端子が０になっていればＴｘＥＮをイネーブルにセットする事で送信可能になります．

2.6.3.4　８２５１のステータス

ステータスの読み取りは，入力ポート２１Ｈ番地から入力する事によって行なえますが，その時読み込んだ各ビットはそれぞれ次の様な意味をもっています．



2.6.3.4.1　DSR

モデム等の制御信号で，データ通信可能な状態であるか否かを示します．

2.6.3.4.2　SYNDET

同期通信の場合のみ使用します．

2.6.3.4.3　FE

有効ストップ・ビットが検出されない場合にセットされ，ERビット・コマンドによってリセットする事ができます．

2.6.3.4.4　OE

次のキャラクタが準備されているにもかかわらず，前のキャクタをCPUが読み出さない場合にセットされ，ERビット・コマンドによってリセットされます．

2.6.3.4.5　PE

パリティ・エラーが検出された場合にセットされ，ERビット・コマンドでリセットされます．

2.6.3.4.6　TxE

8251が，送信データをシリアルに変換して送り出すには，送信バッファの内容をパラレル・シリアル変換回路によってシリアル・データに変換した上で外部に送り出しています．

送信バッファの内容を，パラレル・シリアル変換回路にセットした時点で，送信バッファが空になり，ＴｘＲＤＹがセットされますがこの時点ではデータ全てが外部に送り出されたわけではありません.

全てのデータが外部に送り出されて始めてＴｘＥがセットされ，８２５１が全てのデータ送信を終了した事が確認できます.

2.6.3.4.7　RxRDY

８２５１が，受信したシリアル・データをパラレル・データに変換し終え，ＣＰＵに渡せる状態にある時セットされます.

2.6.3.4.8　TxRDY

このビットが１にセットされている場合には，８２５１の送信バッファが空になっている事を示し，次の送信データをバッファにセットする事が可能になります.

2.7　コントロール・ポート

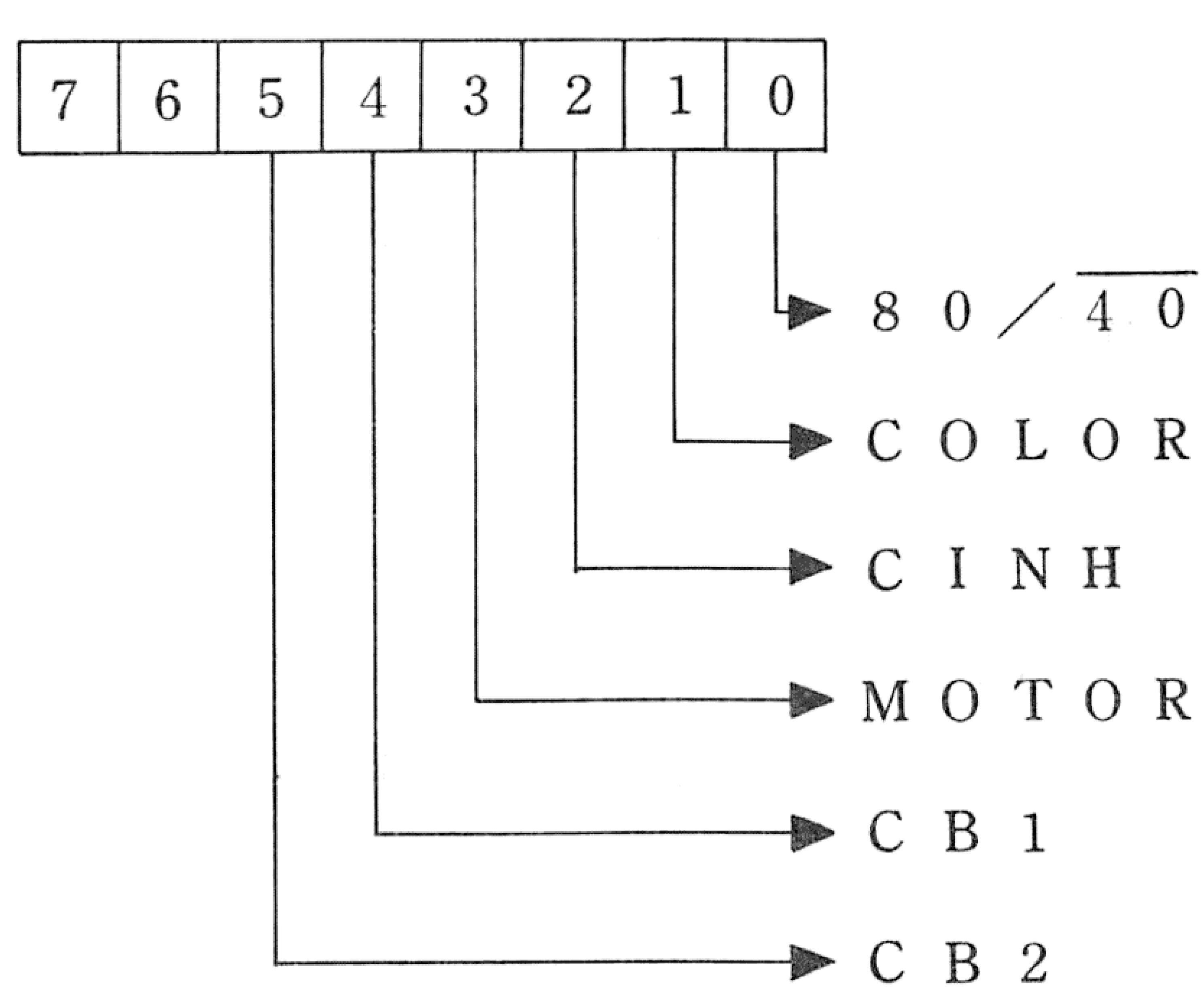
2.7.1　コントロール・ポート１

コントロール１のポートは出力ポートで，各周辺機器のコントロールに使用されます．

Ｉ／Ｏアドレスは，３０Ｈ番地に設定されており，下位４ビットのオーダ・コードはデコードされていませんから，３０Ｈ～３ＦＨ番地を指定すれば全て同じ事になります．

	7	6	5	4	3	2	1	0	
M S B	0	0	1	1					L S B

下記の様な構成により，主にシリアル・インターフェース及び，ＣＲＴのコントロールに使用されます．



2.7.1.1　８０／$\overline{４０}$

ＣＲＴコントローラは，イニシアライズ時に，画面に表示する文字数を指定するようになっていますが，Ｎ－ＢＡＳＩＣは，ＣＲＴコントローラを常に８０桁モードで使用し，８０／４０文字の切り換えは，外部のタイミング回路によって行なっています．

したがって，ビデオ・ラムの容量は表示する文字数に関係なく一定で，４０文字の場合には，１キャラクタおきに表示されます．

この，４０文字モードと８０文字モードとの切り換えを行なうためのデータを出力するためのビットで，１を出力すると８０文字モードが指定され，０を出力すると４０文字モ

ードが使定できます.

ただし，N－BASICの出力ルーチンによって，CRT画面への表示を行なう場合には，N－BASICのワーク・エリアも適当な値に変更しておく必要があります.

2.7.1.2 $\overline{\text{COLOR}}$

カラー・モードと白黒モードを切り換えるために使用するビットで，外部回路のみを設定する事ができます.

実際に，カラー・モードを指定する場合には，このビットを0にリセットすると同時にCRTコントローラもそれぞれの状態にイニシアライズする事が必要です.

2.7.1.3 CINH

CMTインターフェースは，マーク2400Hz，スペース1200HzのFSK方式になっていますが，このビットを0にリセットしておいた場合には，CMTへ出力したデータに対応した信号が出され，1にセットしておけば常に1200Hzの信号が出力されます.

2.7.1.4 MOTOR

本体に内蔵されているCMT用のモータ・リレーをコントロールするためのビットで，1でON，0にすればOFFとなります.

2.7.1.5 CB1，CB2

プログラマブル・コミュニケーション・インターフェイス8251をSIOに使用するかCMTに使用するのかを切り換えるのと同時にCMTの転送速度を指定するために使用します.

PC－8001発表初期のカタログにも，

転送速度：600ボー／300ボー

と，銘記されていた様に，8251では2種類の転送速度を選択する事が可能ですが，残念ながら，N－BASICでは300ボーのサポートがなされていません．もちろんマシン語等を用いれば300ボーも可能です.

ＣＢ２	ＣＢ１	機　　能
0	0	ＣＭＴ　６００ボー
0	1	ＣＭＴ　３００ボー
1	0	ＳＩＯ
1	1	使用禁止

2.7.2　コントロール・ポート２

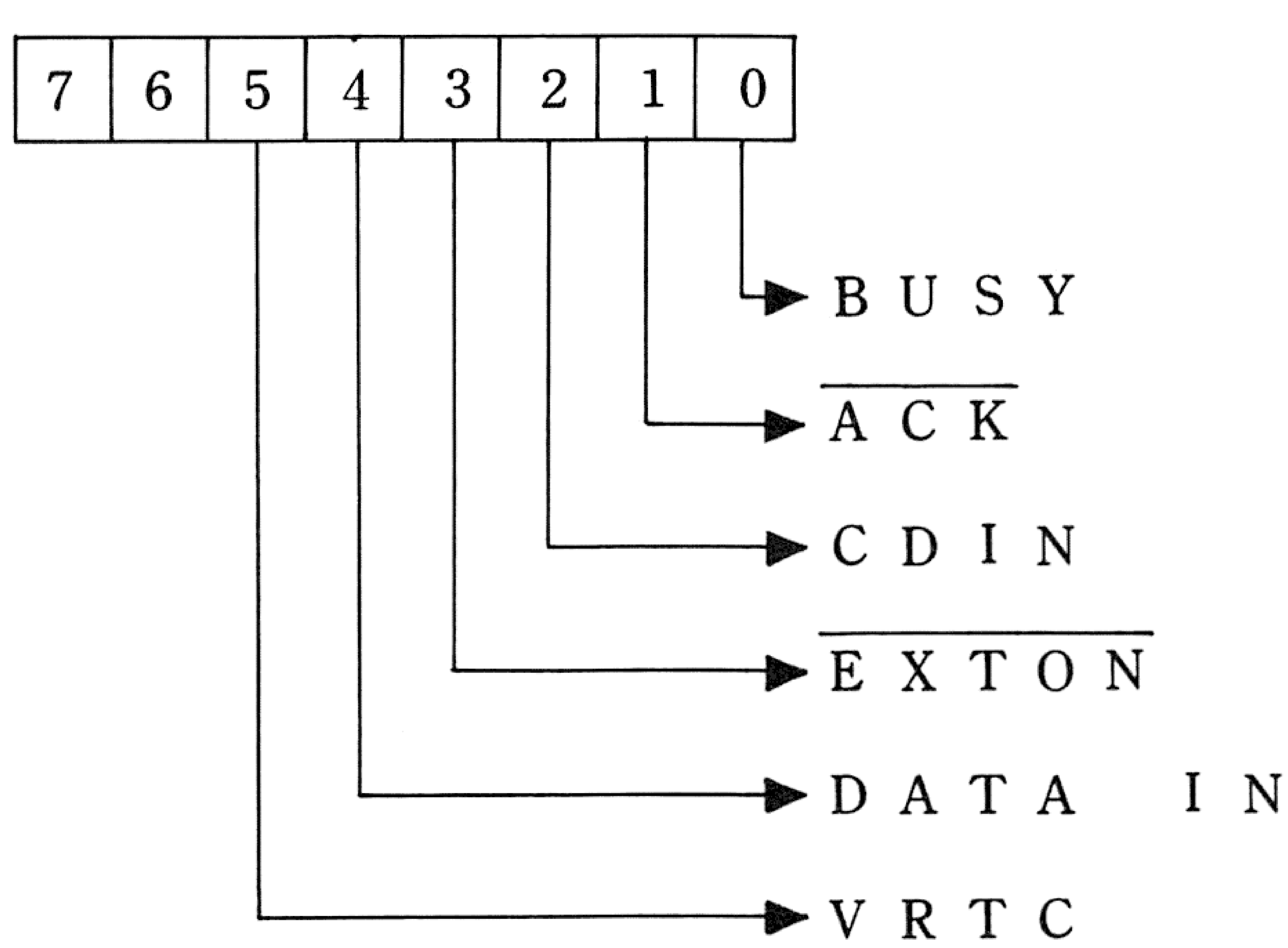
コントロール２のポートは，入出力にわかれており，各周辺機器のコントロールに使用されます．

	7	6	5	4	3	2	1	0	
ＭＳＢ	0	1	0	0					ＬＳＢ

オーダ・コードはデコードされていないためＩ／Ｏアドレスは４０Ｈ番地になります．

2.7.2.1　コントロール２入力ポート

コントロール２の入力ポートは下記の様に構成されています．



2.7.2.1.1　ＢＵＳＹ，$\overline{\mathrm{ACK}}$

それぞれ，プリンタからの$\overline{\mathrm{READY}}$信号，$\overline{\mathrm{ACK}}$信号の検出に使用します．

2.7.2.1.2　ＣＤＩＮ

シリアル・インターフェースＳＩＯのキャリア・ディテクト検出に使用し，ＣＭＴから

のキャリア信号を検出した場合１にセットされます．

2.7.2.1.3 $\overline{\mathrm{EXTON}}$

外部バスに接続されている$\overline{\mathrm{EXTON}}$信号の状態を示しており，ＰＣ－８０３３，ＰＣ－８０１１，ＰＣ－８０１２などが接続されている時の０にリセットされています．

Ｎ－ＢＡＳＩＣではコールド・スタート時に，このビットの値を調べて，ディスク・ユニットのイニシアライズを行なうか否かを決定しています．

2.7.2.1.4 DATA IN

タイマＩＣからの入力を読み込むためのビットで，出力ポートのＣＬＫと同期させて，シリアル・データを受け取ります．

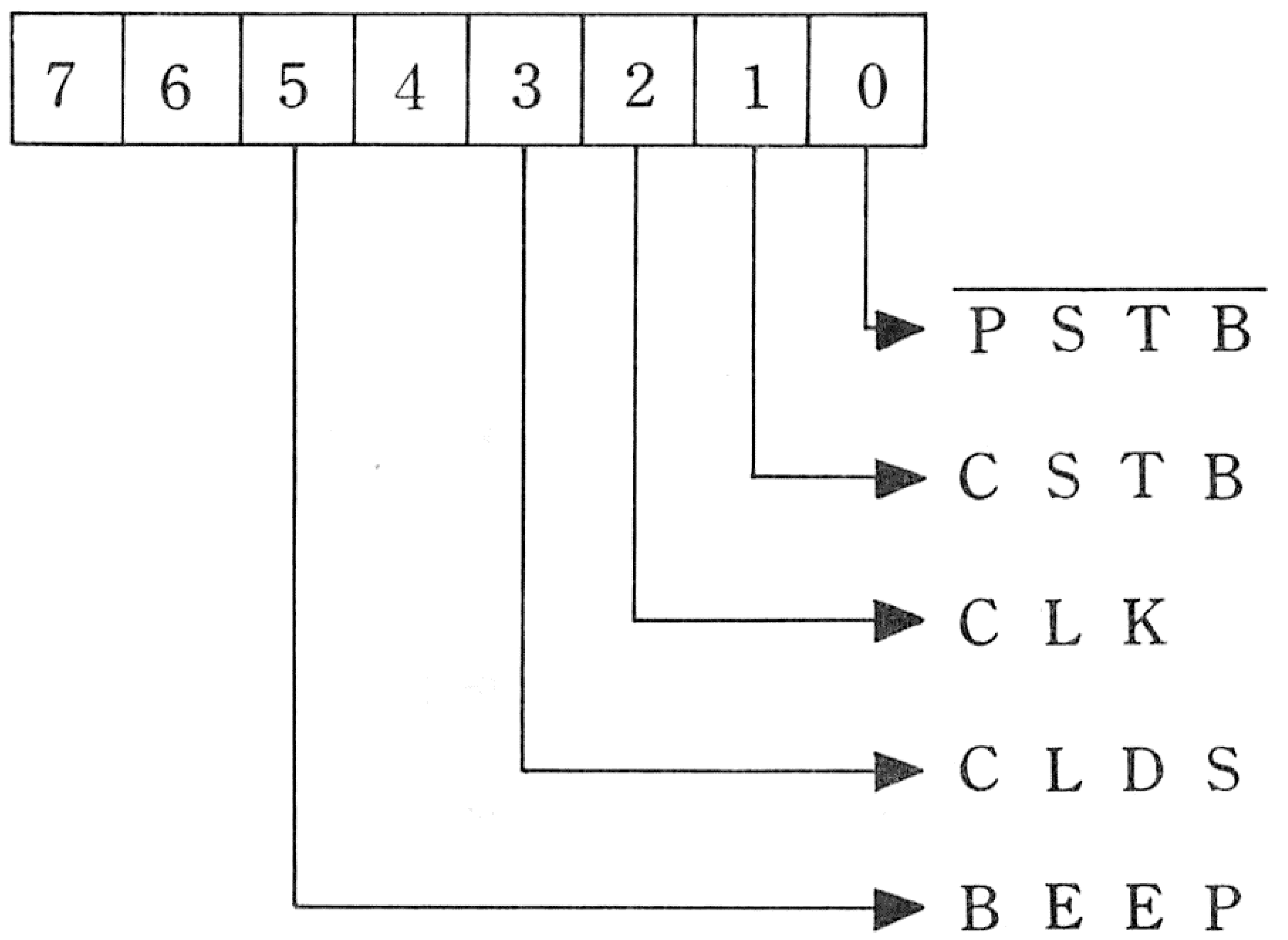
2.7.2.1.5 VRTC

ＣＲＴコントローラの制御に使用します．

０のとき表示およびホリゾンタル・リトレース期間，１のときはバーチカル・リトレース期間です．

2.7.2.2 コントロール２出力ポート

コントロール２の出力ポートは下記の様に構成されています．



2.7.2.2.1 $\overline{\mathrm{PSTB}}$

プリンタへストローブ信号を与えるために使用します．初期状態は１にセットしておきますが，

1→0→1

と変化させる事によって，ストローブ信号を与える事ができます.

2.7.2.2.2　CSTB

タイマＩＣにコマンドをセットするためのストローブ信号を与えるために使用します.

2.7.2.2.3　CLK

タイマＩＣのデータ入出力コントロールのためのクロックを与えます.

2.7.2.2.4　CLDS

ＣＲＴコントローラを初期設定する時点で使用されているビットです.

2.7.2.2.5　BEEP

内蔵スピーカを制御するためのビットで，１にセットする事で２．４ＫＨｚのクロックでスピーカを鳴らすことができます.

回路上では，このビットからの出力信号と２．４ＫＨｚのクロック信号とのＡＮＤをとったものが，スピーカに出力される様になっています.

2.8 CRT

2.8.1 DMAコントローラ

DMAコントローラは，CRTコントローラへビデオ・ラム（VRAM）のデータをDMA転送する目的で使用され，I／Oアドレスは次のようになっています.

MSB	0	1	1	0	A_3	A_2	A_1	A_0	LSB

A_3	A_2	A_1	A_0	機　　能
0	0	0	0	CH－0　DMAアドレス・セット
0	0	0	1	CH－0　ターミナル・カウンタ・セット
0	0	1	0	CH－1　DMAアドレス・セット
0	0	1	1	CH－1　ターミナル・カウンタ・セット
0	1	0	0	CH－2　DMAアドレス・セット
0	1	0	1	CH－2　ターミナル・カウンタ・セット
0	1	1	0	CH－3　DMAアドレス・セット
0	1	1	1	CH－3　ターミナル・カウンタ・セット
1	0	0	0	モード・セット&ステータス・リード

8257はプログラマブルなDMAコントローラで，4チャンネルに対して，アドレス及び転送容量を指定する事ができますが，N－BASICではオート・ロード・モードを使用しているためにチャンネル2しか用いる事はできません.

オート・ロード・モードでは，一垂直画面ごとに自動的に，DMAアドレスのロード及びプリセットが行なわれています.

すなわち，実際に利用するポートのI／Oアドレスは，次のようになります.

Ｉ／Ｏアドレス		機　　　能
６４Ｈ	出力	ＤＭＡアドレス・セット
６５Ｈ	出力	ターミナル・カウンタ・セット
６８Ｈ	出力	モード・セット
６８Ｈ	入力	ステータス・リード

2.8.2　ＣＲＴコントローラ

ＣＲＴコントローラμＰＤ３３０１には，コマンド及びパラメータを設定する事によって，ＣＲＴに関する多くのコントロールを行なわせる事ができます．

Ｉ／Ｏのアドレス構成は次のようになっています．

ＭＳＢ								ＬＳＢ
	０	１	０	０				A_0	

A_0	機　　　能
０	パラメータ設定
１	コマンド設定

すなわち，パラメータ設定の場合は５０Ｈ番地，コマンド設定には５１番地を用いる事になります．

2.8.2.1　スクリーン・フォーマットのイニシアライズ

画面フォーマットのイニシアライズは，ＣＲＴコントローラにリセット・コマンドを与えた後，５バイトに渡るスクリーン・フォーマット用データを出力する事によって行なう事ができます．

この，リセット・コマンドとカーソル設定のためのコマンドは，複数のパラメータを必要とするために，途中でＤＭＡがはじまると表示データとコマンドのパラメータを混同してしまい，正常な動作が不可能になります．これを防ぐためには，リセット・コマンドとカーソル設定コマンドの前後に同期をとらなければなりません．

2.8.2.1.1 スクリーン・フォーマット1

C/B	H	H	H	H	H	H	H

▷DMA動作指定

C/B	モ　ー　ド
0	バースト・モード
1	キャラクタ・モード

▷桁指定

HHHHHHH	桁　　　数
0000000	2桁／行
0000001	3桁／行
～	
0100110	40桁／行
～	
1001110	80桁／行
～	
1111110	
1111111	

2.8.2.1.2 スクリーン・フォーマット2

B	B	L	L	L	L	L	L

▷ブリンキング・タイム指定

BB	カーソル	アトリビュート
00	16	32
01	32	64
10	48	96
11	64	128

▷行数

LLLLLL	行　　　数
000000	1行／画面
000001	2行／画面
～	
010011	20行／画面
～	
011000	25行／画面
～	

2.8.2.1.3 スクリーン・フォーマット3

S	C	C	R	R	R	R	R

▷表示モード

S	モ　ー　ド
0	通常の表示
1	1行おきの表示

▷カーリル表示モード

CC	カ　ー　リ　ル
0 0	アンダ・カーソル
0 1	ブリンキング・アンダ・カーソル
1 0	ブロック・カーソル
1 1	ブリンキング・ブロック・カーソル

▷ライン数

RRRRR	ラ　イ　ン　数
0 0 0 0 0	指定禁止
0 0 0 0 1	指定禁止
0 0 0 1 0	3ライン／文字
0 0 0 1 1	4ライン／文字
～	
0 0 1 1 1	8ライン／文字（20行）
～	
0 1 0 0 1	10ライン／文字（25行）
～	
1 1 1 1 1	32ライン／文字

2.8.2.1.4　スクリーン・フォーマット４

V	V	V	Z	Z	Z	Z	Z

▷垂直帰線幅

ＶＶＶ	垂直帰線幅
０００	１行
∫	
１０１	６行（２０行モードで使用）
１１０	７行（２５行モードで使用）
１１１	８行

▷水平帰線幅

ＺＺＺＺＺ	水平帰線幅
∫	
０００１１	指定禁止
００１００	６文字
∫	
１１１１０	３２文字
１１１１１	３３文字

2.8.2.1.5 スクリーン・フォーマット5

A_T	A_T	S_C	A	A	A	A	A

▷アトリビュート・特殊制御文字モード

A_T A_T S_C	アトリビュート	特殊制御文字
0 0 0	トランスペアレント白黒	有効
0 0 1	ノン・アトリビュート	無効
0 1 0	トランスペアレント・カラー	有効
0 1 1	指定禁止	指定禁止
1 0 0	ノン・トランスペアレント白黒	有効
1 0 1	ノン・トランスペアレント白黒	無効
1 1 0	指定禁止	指定禁止
1 1 1	指定禁止	指定禁止

▷最大アトリビュート数

AAAAA	最大アトリビュート数
00000	1個／行
00001	2個／行
～	
10011	20個／行
～	
11111	32個／行

2.8.2.2 CRTコントローラのアトリビュート

CRT画面上に単にキャクタを表示させるだけであれば，VRAMにキャラクタ・コードをストアするだけで良いのですが，それらの文字にアトリビュート（属性）も設定する事が必要です．

PC－8001が使用しているCRTコントローラ（3301）では，文字コードとアトリビュートを混在して使用するノン・トランスペアレント・アトリビュートと文字コード・エリアとアトリビュート・エリアに分離したトランスペアレント・アトリビュートを使用する事ができ，両者の選択はCRTコントローラを初期設定する時点で行なう事が可能で，アトリビュートを設けない事もできます．

ノン・トランスペアレント・アトリビュートは，文字コードとアトリビュート・コードを同一のVRAM上に混在する方法で両者の区別を最上位ビットによって行なう非常に合理的な方法ではあるのですが，文字コードとしては0～127まで128個のキャラクタ

しか使用する事ができません．Ｎ－ＢＡＳＩＣでは０～２５５まで２５６種のキャラクタを用いるために，トランスペアレント・アトリビュートを採用しています．

2.8.2.2.1 アトリビュート・コード表

BIT		カラーモード		白黒モード
		カラー指定	機能指定	
0	0		ノーマル	ノーマル
	1		シークレット	シークレット
1	0		ノーマル	ノーマル
	1		ブリンク	ブリンク
2	0		ノーマル	ノーマル
	1		リバース	リバース
3	0	機能指定		
	1	カラー指定		
4	0	キャラクタ	ノーマル	ノーマル
	1	グラフィック	オーバー・ライン	オーバー・ライン
5	0	Ｂ信号　ＯＦＦ	ノーマル	ノーマル
	1	Ｂ信号　ＯＮ	アンダー・ライン	アンダー・ライン
6	0	Ｒ信号　ＯＦＦ		
	1	Ｒ信号　ＯＮ		
7	0	Ｇ信号　ＯＦＦ		キャラクタ
	1	Ｇ信号　ＯＮ		グラフィック

ＲＧＢ信号→カラー変換表

信号＼カラー	黒（ブラック）	青（ブルー）	赤（レッド）	紫（マゼンダ）	緑（グリーン）	水（シアン）	黄（イエロー）	白（ホワイト）
Ｇ	ＯＦＦ	ＯＦＦ	ＯＦＦ	ＯＦＦ	ＯＮ	ＯＮ	ＯＮ	ＯＮ
Ｒ	ＯＦＦ	ＯＦＦ	ＯＮ	ＯＮ	ＯＦＦ	ＯＦＦ	ＯＮ	ＯＮ
Ｂ	ＯＦＦ	ＯＮ	ＯＦＦ	ＯＮ	ＯＦＦ	ＯＮ	ＯＦＦ	ＯＮ

2.8.2.2.2 ＶＲＡＭとアトリビュート・エリアの構成

ＶＲＡＭを見てもわかるように，一画面が１２０バイト×２５行で構成されています．１２０バイトの内，前半８０バイトが純粋な意味のＶＲＡＭで残りの４０バイトがアト

リビュート・エリアですから１行につき４０バイト分のアトリビュート指定が行なえますが，実際には2バイトで１組のアトリビュート指定を行ないますから，１行あたりのアトリビュート数は最大で２０個まででです.

このアトリビュート指定用データの第１バイト目は，１行中の何番目の文字に対して指定を行なうかを意味し，画面の左端が0になりますが４０文字モードの場合には2倍した値を与えてやらねばなりません.

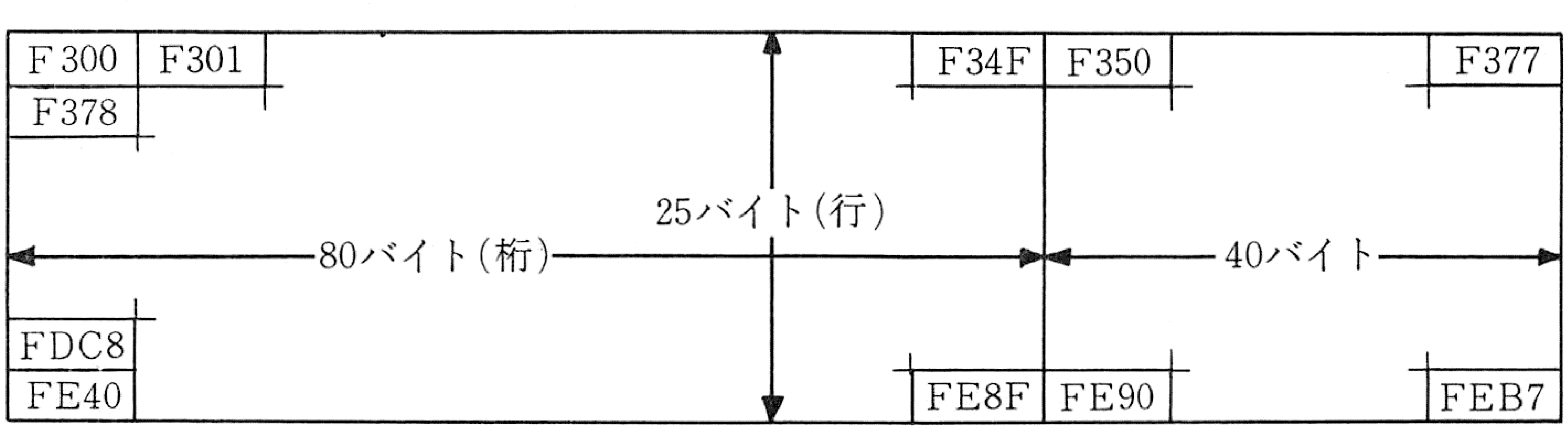
第2バイト目が実際のアトリビュートを指定するためのアトリビュート・コードになります.

白黒モード時においては，アトリビュートも，リバースやブリンク等の機能のみを指定すれば良く問題も無いと思いますが，カラー・モードの場合には機能指定とカラー指定の2種類が必要なため多少面倒です.

この，機能指定とカラー指定はアトリビュート・データの第3ビットによって区別され両者は全く独立して指定する事ができます．つまり，一度機能指定を行えば途中にカラー指定が存在しても次の機能指定まではじめに設定した機能が継続し，なおかつカラー指定も有効に動作するのです．逆に，一度カラー指定を行なえば，いくら機能が変化してもカラーは変化しません.

これらの機能を有効に使用すれば，Ｎ－ＢＡＳＩＣのＣＯＬＯＲ文とＰＲＩＮＴ文だけでは不可能だった，カラー・モードでの１文字ブリンクやリバースなどが比較的簡単に実行できますが，当然アトリビュート指定に比例して，アトリビュート•エリアを使います.

VRAMとCRT画面の関係



2.8.2.2.3 アトリビュート・エリアの初期値

通常ＶＲＡＭは，キャラクタ・モードのとき２０Ｈで，グラフィック・モードのときは0でクリアします．それではアトリビュート・エリアのクリアはどのようにして行うのでしょうか？

第１バイト目の列アドレスは１行あたりの文字数より大きい値でクリアする事になって

いますが例外として一番最初の列アドレスには0を入れておかなければなりません．ちなみに，N－BASICでは50Hでクリアしています．

第2バイト目のアトリビュート・コードは白黒モードのときは0を，カラー・モードのときにはカラー指定にしてベース・カラーとするべきコードをセットしておきます．例えばベース・カラーを白にする場合はE8H，緑にする場合には88Hを使用します．又，グラフィックを使用するには，グラフィックのビットをセットしてクリアするなど，場合によっていろいろなデータを用いる事ができます．

2.8.2.2.4 アトリビュート・コントロールの実際

当然の事ながらアトリビュート・エリアに適切なデータを入れる事によってCRT画面に表示する文字のアトリビュート（属性）を指定する事ができます．

もちろん，直接アトリビュート・エリアの内容を変更して，アトリビュートのコントロールを行なっても良いのですが，N－BASIC内にも凡用性のあるコントロール・ルーチンが用意されています．一般に高速性等を求めない限りはそれらのサブ・ルーチンを利用してアトリビートをコントロールした方が容易でしょう．

2.8.2.2.4.1 プリント・ルーチン使用時のコントロール

N－BASIC内には，文字や数値等をCRT画面に表示するために多くのサブルーチンを持っていますが，それら全てのサブルーチンでは，EA5BH番地のデータを表示時のアトリビュートとして，そのままアトリビュート・エリアに送り込んでいます．

つまり，N－BASIC内のシステム・サブルーチンを利用してCRT画面への表示を行なう場合に限っては，EA5BH番地の値を変える事によって簡単にアトリビュート指定ができるのです．

EA5BH番地に入れる内容は，アトリビュート・エリアに送り込むと同じ値でなくてはいけませんから，例えばカラー・モード時で白の時E8H，緑の時88H，アンダー・ラインを付けたければ20Hとなります．

しかし，この方法では，CRT画面への表示ルーチンを利用するためキャラクタにしかアトリビュートを指定する事ができず，ドット・グラフィックなどを扱う事はできません．

2.8.2.2.4.2 座標を指定したコントロール

CRT画面上の任意の座標をした上で，1キャラクタ分だけのアトリビュートを設定する事も04F8H番地からのシステム・サブルーチンで行なう事ができます．

このサブルーチンに与える入力パラメータは,

Hレジスタ ← CRT横座標

Lレジスタ ← CRT縦座標

Aレジスタ ← アトリビュート・コード

で, 座標（H, L）に対応したVRAM上の1キャラクタ分を, Aレジスタのアトリビュート・コードに設定する働らきを持ち, キャラクタのみではなく, ドット・グラフィック等の指定を行なうこともできます.

ただし, Hレジスタ対に与える座標は, N－BASICのLOCATEで使用するパラメータに1を加えたもので, 画面左上端は（0, 0）でなく（1, 1）になります.

また, このサブルーチンでは, CRT画面表示キャラクタ数や, カラー／白黒モードの判別も自動的に行なわれます.

座標を指定した, アトリビュート・コントロール・サブルーチンの利用例として, 現在のカーソル位置を中心とした3×3のエリア内を, 反転（リバース）キャラクタに設定するサブルーチンを示します.

```
                ;******************************
                ;
                ;  TEST SET ATTRIBUTE ROUTINE
                ;
                ;******************************
                ;
0004            REVCOD:EQU  04H          ; ATTRIBUTE CODE OF REVERSE
04F8            CTLATR:EQU  04F8H        ; CONTROL ATTRIBUTE SUBROUTINE
EA63            CUSPOS:EQU  0EA63H       ; CURSOR POSITION ON SCREEN
                ;
                       ORG  9000H
                ;
9000 2A63EA            LD   HL,(CUSPOS)
9003 3E04              LD   A,REVCOD
9005 CD2990            CALL CSTATR
9008 24                INC  H
9009 CD2990            CALL CSTATR
900C 2D                DEC  L
900D CD2990            CALL CSTATR
9010 25                DEC  H
9011 CD2990            CALL CSTATR
9014 25                DEC  H
9015 CD2990            CALL CSTATR
9018 2C                INC  L
9019 CD2990            CALL CSTATR
901C 2C                INC  L
901D CD2990            CALL CSTATR
9020 24                INC  H
9021 CD2990            CALL CSTATR
9024 24                INC  H
9025 CD2990            CALL CSTATR
9028 C9                RET
                ;
9029 F5         CSTATR:PUSH AF
902A E5                PUSH HL
902B CDF804            CALL CTLATR
902E E1                POP  HL
902F F1                POP  AF
9030 C9                RET
                ;
```

2.9　ディスク

本書に記載した事項の中で，ディスク・ユニットおよびＤｉｓｋ－ＢＡＳＩＣに関するものは，特別の指定がない限り，ＮＥＣのＰＣ－８０３１を，同じくＮＥＣの，Ｄｉｓｋ　ｖｅｒｓｉｏｎ　［１４－Ｎｏｖ－１９８０］によってドライブするシステムを想定してありますので御了承ください．

2.9.1　ＩＰＬについて

Ｎ－ＢＡＳＩＣは，電源投入時等のコールド・スタートした場合に，ミニ・ディスク・ユニットが接続されているか否かを調べ，接続されている場合には，ディスケットからＩＰＬをロードします．

ＩＰＬはディスケット上のトラック０セクタ１に置いてあり，そのセクタ２５６バイトの内容がＰＣ－８００１のＣ０００Ｈ番地にロードされて自動的に実行されます．

ＩＰＬの実行によって，ディスケット上のトラック０セクタ２からトラック２セクタ１６までに置かれているディスク・コードがロードされる様になっています．

つまり，トラック０セクタ１の２５６バイトに書かれているプログラムは，Ｎ－ＢＡＳＩＣのコールド・スタート時には無条件にＣ０００Ｈ番地にロードされて実行される事になり，この機能を有効に使用すれば，ディスクのオート・スタートなどの多くの応用が考えられます．

2.9.2　ディスク・ファイルの管理

ＰＣ－８０３１を扱うための，Ｄｉｓｋ－ＢＡＳＩＣでは，８つのセクタを１つのクラスタとして，ファイルの管理を行なっています．

１つのクラスタは２Ｋバイトになり，トラック０のセクタ１～８までをクラスタ０，トラック３４のセクタ９～１６をクラスタ６９として，１枚のディスケットは７０のクラスタによって構成されています．

このため，どのようなファイルも全て２Ｋバイトの整数倍の領域を取る事になり，例えば，４０９７バイトのファイルは，

　　４０９７÷２０４８＝２　余り　１

で，３クラスタの領域を使用する事になります．この例では最後のクラスタは１バイトしか用いていませんが，それでもしっかりと１クラスタ分を確保します．

2.9.2.1　DIRとFAT

Ｄｉｓｋ―ＢＡＳＩＣは，各ファイルと，クラスタの照合を行なうために，ディレクトリとＦＡＴと呼ばれる２つの表を管理しています．

ディレクトリには，ディスケット内に収納されている全てのファイルの名称，属性，先頭の収納場所が記憶されており，ＦＡＴは各クラスタの使用有無や連絡状態を記憶しています．

2.9.2.1.1　DIR

ディレクトリは，トラック１８のセクタ１～１２に取られて各ファイルに対して１６バイトが割り当てられ，次表のような意味をもっています．

１枚のディスケットに対して１２のセクタをディレクトとして使用しますので，ディレクトリには最大１９２個のファイルが登録できる事になります．

バイト	意　　味
0～5	ファイル名 先頭バイトがＦＦＨのとき，そのディレクトリは未使用であり，０のとき削除されたファイルである事を示す。
6～8	拡張子
9	属性 ００Ｈ：　アスキー形式 １０Ｈ：　アスキー形式　Ｐ ２０Ｈ：　アスキー形式　Ｒ ８０Ｈ：非アスキー形式 ９０Ｈ：非アスキー形式　Ｐ Ａ０Ｈ：非アスキー形式　Ｒ
１０	ファイルが収納されている先頭のクラスタ・ナンバ
11～15	未使用

2.9.2.1.2　ＦＡＴ

ＦＡＴは各クラスタの使用状態を示す表でトラック１８のセクタ１４～１６に書き込まれており，ＭＯＵＮＴ時にＰＣ－８００１内のＲＡＭ上にコピーされます．

安全のため，セクタ１４～１６の内容は全て同じで，３つのセクタの内容が一致しないディスケットは使用できません．

また，それぞれのセクタの０～６９バイトが０～６９クラスタに対応しており，各クラスタの使用状況を示しています．

デ　ー　タ	意　　　味
００Ｈ～４５Ｈ	使用中．連続したクラスタの一部であり後続したクラスタを持つ．値が後続するクラスタ・ナンバを示す。
Ｃ１Ｈ～Ｃ８Ｈ	使用中．連続したクラスタの最後のクラスタであり下位4ビットの内容が，そのクラスタで実際に使われているセクタ数を示す。
ＦＥＨ	システムが予約ずみのクラスタで，ＩＰＬ，Ｄｉｓｋコード，ディレクトリ，ＦＡＴ等に使用されているクラスタ。
ＦＦＨ	未使用。自由に使用できる事を示す。

2.9.2.2　ＩＤ

ＩＤとしては，トラック１８のセクタ１３の２５６バイトのみが割り当てられていますが，Ｄｉｓｋ－ＢＡＳＩＣでは，先頭の１バイトのみを使っていて，２５５バイトは残っています．

この１バイトは，ディスケット１枚全てに通用する属性を示しており，ディレクトリに書き込む属性と同じものです．

2.9.3　ストップ・リセット

Ｄｉｓｋ－ＢＡＳＩＣを使用している時に何らかの理由によってリセットをかけなけれ

ばならなくなる事は意外と多いものです．特にマシン語のプログラムを扱って無限ループに入った場合などはどうしてもリセットをかけなければなりません．

このような場合にＳＴＯＰリセットによって，Ｎ－ＢＡＳＩＣをコールド・スタートさせてしまうと，ディスク・ユニットが切り離されて，ＭＯＵＮＴやＲＥＭＯＵＥ等，ディスクをアクセスする事が不可能になりますがこれを「デス・ドライブ」と呼びます．これは，ＲＥＳＥＴ信号がほとんど全てのＬＳＩのピンに直接つながっているためほとんどのＬＳＩをリセットしてしまうためでμＰＤ８２５５も例外ではありません．

この，デス・ドライブの状態からぬけ出すためには，００Ｄ２Ｈ番地からのコントローラ初期化サブルーチンを実行させるか，出力ポートＦＦＨ番地に９１Ｈを出力して，μＰＤ８２５５のモード設定を行なえば良いのです．一般には，モニタから，

ＧＤ２$^{C}_{R}$

を行なうのが良いでしょう．

2.9.4 エスケープ・リセット

逆に，Ｄｉｓｋ－ＢＡＳＩＣの使用中に，Ｄｉｓｋ－ＢＡＳＩＣを切り離してＮ－ＢＡＳＩＣのみを動かしたい場合には，ＥＳＣキーを押しながらリセットをかける事が有効です．

2.9.5 ＰＣ－８０３１ ミニ・ディスク・ユニット

ＰＣ－８０３１はインテリジェント・タイプのミニ・フロッピ・ディスク・ユニットであり，ＣＰＵとして，ザイログのＺ８０ＡコンパチブルなＮＥＣのμＰＤ７８０を使用しています．

内部にはファーム・ウェアを焼き込んだ２ＫバイトのＲＯＭと１６ＫバイトのＤＲＡＭを有し，コントローラとして高機能なフロッピ・ディスク・コントローラであるμＰＤ７６５を，入出力ポートとしてμＰＤ８２５５を使用しており，３線ハンドシェイク方式で本体側のμＰＤ８２５５と接続されています．

このためＰＣ－８０３１自体が独立したコンピュータとしての機能を持ち，本体側はＰ

C－８０３１に対してコマンドやデータを送るのみで，実際のディスケットへのアクセスは全てＰＣ－８０３１に任せる事になります．この方法では，速度の点では不利になりますが，本体側のＣＰＵに負担がかからなくなります．

2.9.6　ディスクとのインターフェイス

８２５５は，マイクロ・コンピュータ用としては比較的一般的な，プログラマブル汎用Ｉ／Ｏデバイスで，８ビットの入出力ポート３組が内蔵されており，プログラムによって

▷データ入力

▷データ出力

▷ステータス信号入力

▷コントロール信号出力

それぞれに用いる事ができます．

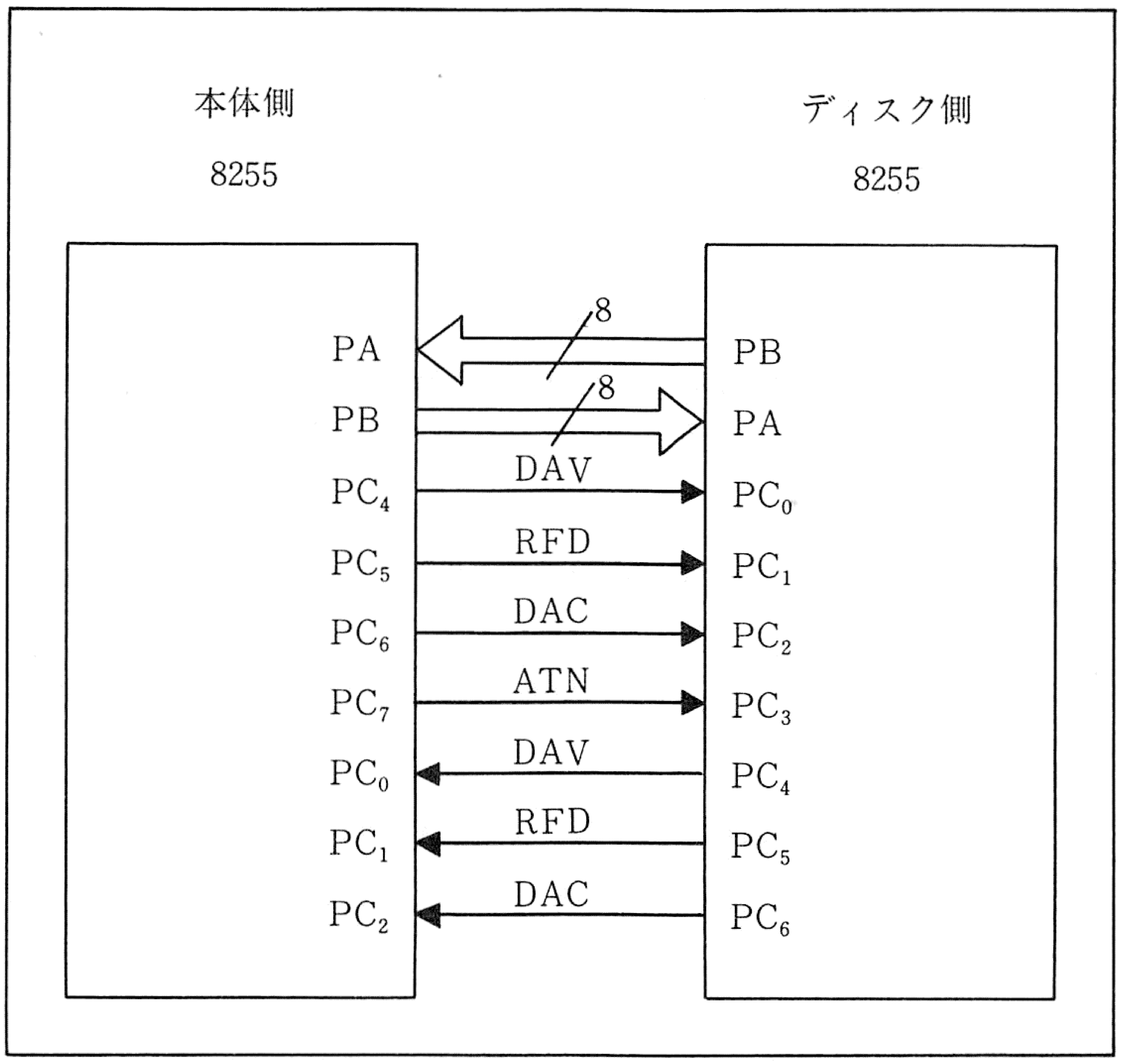
また，３組の入出力ポートは，グループＡ，Ｂに分かれており，別々のモードで使用する事ができますが，これらのモードは８２５５のコントロール・ワードを設定する事によって選ぶ事ができます．

モード０：単純な入出力ポート．

モード１：コントロール信号，ステータス信号による制御を伴う入出力ポート．

モード２：双方向データを扱う入出力ポート．

８２５５では最初にモード設定を行ない，どのポートを入／出力用に使用するかを決定します．このモード設定はＦＦＨ番地のポートに８ビットのコントロール・ワードをセットする事によって行ないますが，コントロール・ワードは，最上位ビットによってモード設定かポートＣのビット・セット／リセットを行なうかを指定し，モード設定はグループＡ，Ｂに対してデータの組み合わせによって行ないます．



本体側のμＰＤ８２５５とＰＣ－８０３１のμＰＤ８２５５の各ポートは上図のように接続され，本体側のポートＢがＰＣ－８０３１への出力データ，ポートＡがＰＣ－８０３１からの入力データ，ポートＣの下位４ビットがＰＣ－８０３１への制御信号の出力，上位４ビットが制御信号の入力用に使われています．つまり，μＰＤ８２５５をモード０で使用している事になり，このモードは，出力ポートＦＦＨ番地に９１Ｈを出力する事によって実現されます．

2.9.7 制御信号

実際にＰＣ－８０３１へのコマンドの送信やデータの送受信は，４種の制御信号をプログラムによって制御する事で行ないます．

ＡＴＮ（Ａｔｔｅｎｔｉｏｎ）

次に送る信号がコマンドである事を示します．

ＲＦＤ（Ｒｅａｄｙ　Ｆｏｒ　Ｄａｔａ）

データやコマンドを受け取る準備ができている事を示します．

ＤＡＶ（Ｄａｔａ　Ｖａｌｉｄ）

データが有効であるため受信をうながすための信号です．

ＤＡＣ（Ｄａｔａ　Ａｃｃｅｐｔｅｄ）

データを受身した事を示し，ＤＡＶのリセットをうながします．

2.9.8 コマンドおよびデータのプロトコル

ＰＣ－８０３１にコマンドを送るためには最初にＡＴＮをセットして次に送る信号がコマンドである事を示します．

次に，ＲＦＤをチェックして準備ができていればＡＴＮをリセットしてコマンドを送りそれが有効である事を示すためにＤＡＶをセットしてＤＡＣがセットされるのを待ちます．

ＤＡＣがセットされればＤＡＶをリセットして送信を終わります．

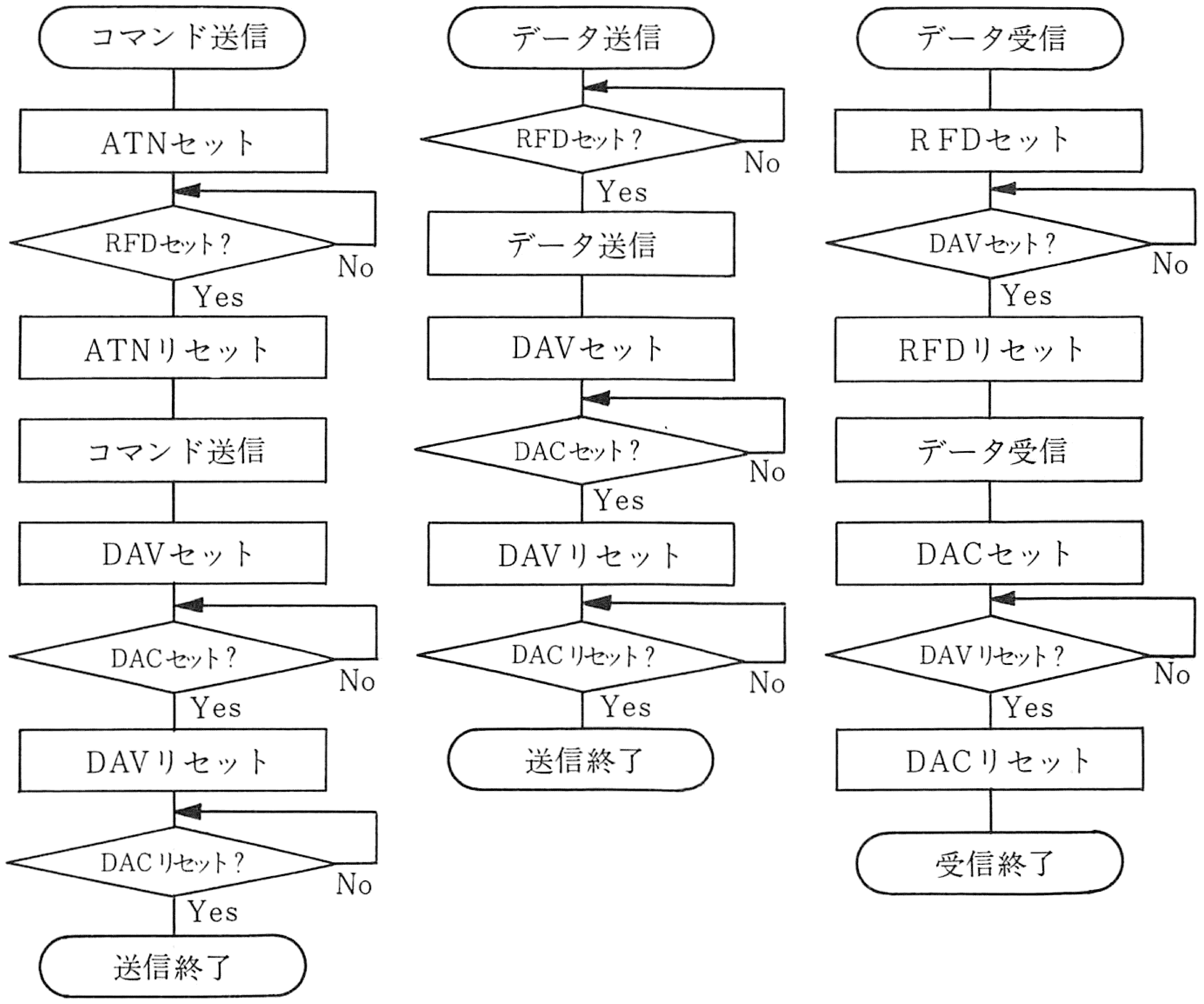
データの送信もコマンドの場合とほとんど同じですが，ＡＴＮのセットとリセットを行なわない部分だけが異なります．

逆に，ＰＣ－８０３１からのデータを受信する場合には最初にＲＦＤをセットしてデータの受信準備ができている事を示しＤＡＶがセットされるまで待ってからデータを受信します．

後は，ＤＡＣをセットしてデータを受信した事を示し，ＤＡＶがリセットされるまで待ってからＤＡＣをリセットして終了します．

以上が，ＰＣ－８０３１との間での１バイト・データの送受信及び，コマンドの送信方法ですが，説明だけではわかりにくいと思われますので，実際のフロー・チャート例をあげておきます．

なお，送受信するデータが２バイト以上に渡る場合でも同様にしてプロトコルを行なう事ができます．



2.9.9　ディスク入出力ルーチンの使用方法

Ｄｉｓｋ―ＢＡＳＩＣ関係のコマンドや関数などは全て，ＲＡＭ上のワーク・エリアにあるジャンプ・テーブルを介してディスク・コード内に飛び込んでいます．

しかし，全てがディスク・コードの中で処理されているのでは無く，最も重要なディスク・ユニットへの入出力は全て，ＲＯＭ上の９９Ｈ番地からに置かれている，ディスク入出力サブルーチンによって行なわれているのです．この入出力ルーチンは非常に便利なサブルーチンで，ディスク・ユニットへの入出力に関する事ならほとんど全てをこなしてしまいます．

もちろん、ＲＡＭ上にディスク・コードが存在しない場合にも使用できますし，入出力の行ない方も簡単なのですが，ディレクトリやＦＡＴまでを自動的に書き換えてくれるわけではありません．エラー処理なども十分ではありませんから，よく注意してデータ等を壊さないように活用してください。

９９Ｈ番地からのサブルーチンでは，ＰＣ―８００１のメモリ上からディスケット上へのデータ・ライト，逆にディスケットからメモリへのデータ・リード，及びベリファイをセクタ単位で行なう事ができます．

サブルーチン・コール時には次の入力パラメータを正確に与える事が必要です．

Ａレジスタ　　⟵　入出力実行セクタ数
Ｂレジスタ　　⟵　トラック・ナンバ
Ｃレジスタ　　⟵　セクタ・ナンバ
ＤＥレジスタ対　⟵　入出力開始アドレス

ＥＤＣ５Ｈ番地　⟵０
ＥＤＣ６Ｈ番地　⟵　ドライブ・ナンバ（０－３）

ライト，リード，ベリファイの区別はフラグで行ないます．

ＣＹ　⟵　１：　データ・ライト
Ｚ　　⟵　０：　データ・リード
Ｚ　　⟵　１：　データ・ベリファイ

又，エラー発生ごとに，ＥＤＣ５Ｈ番地の値をインクリメントして行き，エラーが１０回を越えると，

Ｄｉｓｋ　Ｉ／Ｏ　ｅｒｒｏｒ

が出力されます．エラーが発生した事は，キャリー・フラグ（ＣＹ）がセットされている事でも確認する事ができます．

次に，９９Ｈ番地の使用例として，Ｎ－ＢＡＳＩＣのインプット・バッファであるＥＣ９６Ｈ～ＥＤ９５Ｈ番地のデータを，ドライブ１のトラック０，セクタ１に書き込むサブルーチン例を挙げておきます．

```
ＬＤ　　ＢＣ，０００１Ｈ
ＬＤ　　ＤＥ，ＥＣ９６Ｈ
ＸＯＲ　Ａ
ＬＤ　　（ＥＤＣ５Ｈ），Ａ
ＬＤ　　（ＥＤＣ６Ｈ），Ａ
```

```
INC  A
SCF
CALL 0099H
RET
```

2.9.10 PC−8031メモリ・マップ

アドレス	内容
0000H 〜 07FFH	PC−8031制御ファーム・ウェア2Kバイト
0800H 〜 3FFFH	空き領域
4000H 〜 4FFFH	ライティング・バッファ4Kバイト
5000H 〜 5FFFH	リーディング・バッファ4Kバイト
6000H 〜 7FEFH	フリー・エリア7.85Kバイト
7FF0H 〜 7FFFH	ワーキング・エリア&スタック・エリア

2.9.11　ＰＣ－８０３１のコマンド

2.9.11.1　[０]イニシアライズ・コマンド

つながれている全てのドライブのモータをスタートし，コントローラをイニシアライズするためのコマンドでパラメータはありません．

▷コマンド・コード０

2.9.11.2　[１]ライト・データ・コマンド

本体側より送るデータを，ＰＣ－８０３１内の指定ドライブの指定位置にかき込むためのコマンドで，いくつかのセクタをまとめて書く事ができますが，一度には１つのトラックしかアクセスできません．

▷コマンド・コード１

▷書き込みセクタ数（１～１６）

▷書き込みドライブ・ナンバ（０～３）

▷書き込みトラック・ナンバ（０～３４）

▷書き込み開始セクタ・ナンバ（１～１６）

▷セクタ数×２５６バイトのデータ群

2.9.11.3　[２]リード・データ・コマンド

ＰＣ－８０３１内の指定するドライブ，トラック，セクタから指定セクタ数のデータを読み，ＰＣ－８０３１の５０００Ｈ番地からにあるリーディング・バッファに格納するコマンドで，通常は，センド・データ・コマンドと共に使用されます．

▷コマンド・コード２

▷読み出しセクタ数（１～１６）

▷読み出しドライブ・ナンバ（０～３）

▷読み出しトラック・ナンバ（０～３４）

▷読み出し開始セクタ・ナンバ（１～１６）

2.9.11.4 ［3］センド・データ・コマンド

ＰＣ－８０３１内の５０００Ｈ番地からにあるリーディング・バッファに読み込んだデータを本体側に転送するコマンドですが，このコマンドを実行する以前には，必ずリード・データ・コマンドが実行されていなければなりません．

▷コマンド・コード３

◁リード・データ・コマンドで指定したセクタ数×２５６バイトのデータ

2.9.11.5 ［4］コピー・コマンド

指定するドライブ，トラック，セクタから任意のドライブ，トラック，セクタへデータを転送するためのコマンドで，ＮＥＣのディスケットのバック・アップ・プログラムは，このコマンドを利用しています．

▷コマンド・コード４

▷コピー・セクタ数（１～１６）

▷ソース・ドライブ・ナンバ（０～３）

▷ソース・トラック・ナンバ（０～３４）

▷ソース開始セクタ・ナンバ（１～１６）

▷コピー先ドライブ・ナンバ（０～３）

▷コピー先トラック・ナンバ（０～３４）

▷コピー開始セクタ・ナンバ（１～１６）

2.9.11.6 ［5］フォーマット・コマンド

指定するドライブのディスケットをフォーマッティングするためのコマンドでトラックおよびセクタをわり当て，実際にはディスケット全体にＦＦＨを書き込みます．この段階でも入出力を行なう事は可能ですが，Ｄｉｓｋ－ＢＡＳＩＣでファイルを作成するためにはフォーマット・プログラムなどによって第２レベルのフォーマッティングを行ない，ディレクトリやＦＡＴのイニシアライズを行なわなければなりません．

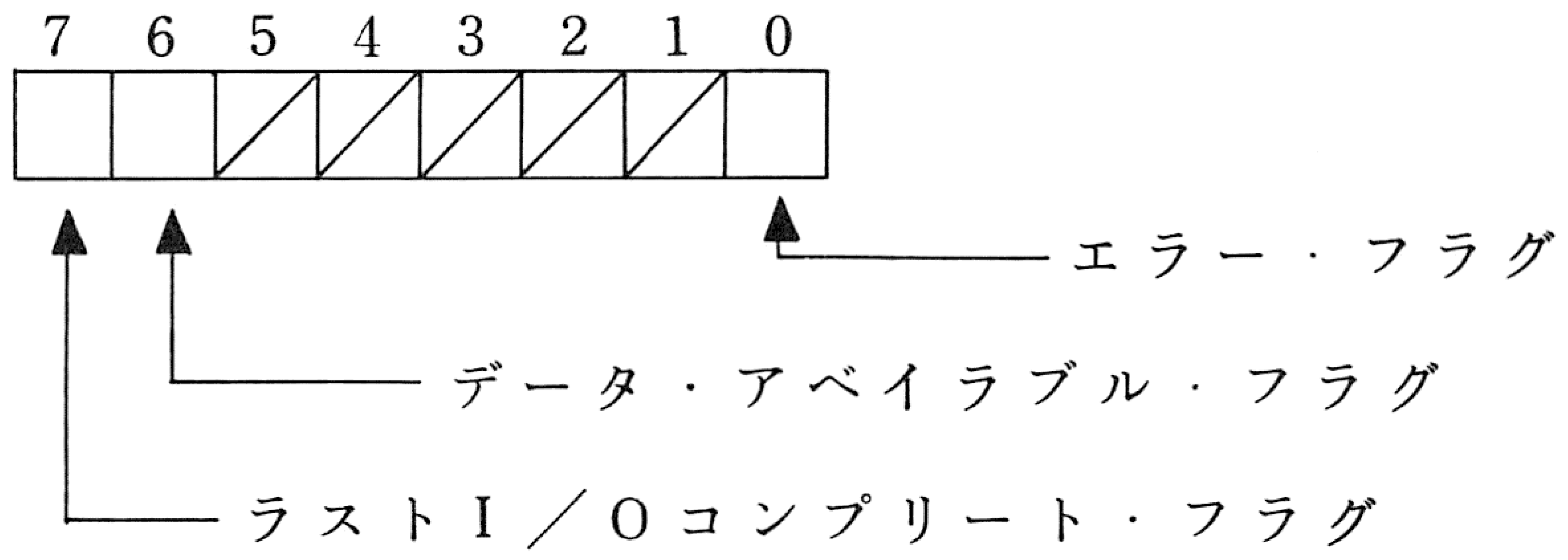
▷コマンド・コード５

▷ドライブ・ナンバ（０～３）

2.9.11.7 [6]センド・レザルト・ステータス・コマンド

このコマンドは，ＰＣ－８０３１の実行結果を調べるためにステータス・バイトを返すためのコマンドです．

レザルト・ステータス・バイト



データ・アベイラブル・フラグがバッファ内にデータがあるかどうかを示し，これが0にリセットされている時にセンド・データ・コマンドを実行すると無限ループに入ってしまいます．第7ビットは常に1で未使用です．

エラー・フラグはエラー発生を表わします．

▷コマンド・コード6

◁レザルト・ステータス・バイト

2.9.11.8 [7]センド・ドライブ・ステータス・コマンド

各ドライブの接続状況を示すステータス・バイトを返すためのコマンドです．

上位4ビットは各ドライブがイニシアライズされたオンライン状態であるかどうかを示し，下位4ビットはディステットがセットされているかどうかを示しますが，ミニ・フロッピ・ディスク・ドライブにはディスケットの有無を調べるためのセンサがないため，下位4ビットは常に1にセットされています．

▷コマンド・コード7

◁ドライブ・ステータス・バイト

2.9.11.9 [11]トランスミット・コマンド

ＰＣ－８０３１内の指定アドレスから指定バイト数のデータを，本体側に転送するためのコマンドです．

▷コマンド・コード１１

▷データ開始アドレス上位８ビット

▷データ開始アドレス下位８ビット

▷転送バイト数上位８ビット

▷転送バイト数下位８ビット

◁指定バイト数のデータ群

2.9.11.10 [12] レシーブ・コマンド

本体側より転送するデータを，ＰＣ−８０３１内の指定アドレスに格納するためのコマンドです．格納アドレスは，６０００Ｈ〜７ＥＦＦＨ番地のフリー・エリアを用いる事が望ましいのですが，ライト，リード，コピーなどの各コマンドを使用していない時には，４０００Ｈ〜５ＦＦＦＨ番地を用いる事も可能です．

▷コマンド・コード１２

▷メモリ先頭アドレス下位８ビット

▷メモリ先頭アドレス上位８ビット

▷転送バイト数上位８ビット

▷転送バイト数下位８ビット

▷転送バイト数のデータ群

2.9.11.11 [13] エクシキュート・コマンド

ＰＣ−８０３１のＣＰＵに対して指定するアドレスにジャンプさせる事のできるコマンドで，ＰＣ−８０３１を本体側とは独立した一つのコンピュータとして機能させる事ができます．

一般には不用意に用いると暴走しますので十分な注意の上で使用してください。

まず，レシーブ・コマンドなどで，ＰＣ−８０３１のメモリ上にマシン語のイメージをつくっておいた上で，このコマンドを用いる事が多いと思います．

▷コマンド・コード１３

▷ジャンプ・アドレス上位８ビット

▷ジャンプ・アドレス下位８ビット

2.9.11.12 [14] ロード・データ・コマンド

指定するドライブ，トラック，セクタから指定セクタ数のデータを読み，ＰＣ－８０３１内の指定アドレスに収納するためのコマンドで，トランスミット・コマンドと共に使用する事によってディスケット上のデータを本体側に転送する事ができます．

ロード・データ・コマンドの実行後には，センド・レザルト・ステータス・コマンドによってエラーの有無を確認する事が必要です．

▷コマンド・コード１４

▷ロード・セクタ数（１～１６）

▷ロード・ドライブ・ナンバ（０～３）

▷ロード・トラック・ナンバ（０～３４）

▷ロード開始セクタ・ナンバ（１～１６）

▷収納開始アドレス上位８ビット

▷収納開始アドレス下位８ビット

2.9.11.13 [15] セーブ・データ・コマンド

ロード・データ・コマンドとは逆に，ＰＣ－８０３１内の指定アドレスから指定セクタ数のデータを，指定のドライブ，トラック，セクタに書き込むためのコマンドですが，実行後はセンド・レザルト・ステータス・コマンドによってエラーの有無を確認しなければなりません．

▷コマンド・コード１５

▷セーブ・セクタ数（１～１６）

▷セーブ・ドライブ・ナンバ（０～３）

▷セーブ・トラック・ナンバ（０～３４）

▷セーブ開始セクタ・ナンバ（１～１６）

▷データ先頭アドレス下位８ビット

▷データ先頭アドレス上位８ビット

2.9.11.14 [16] エクステンション・コマンド

ＰＣ－８０３１をディスケットから読み込んだＩＰＬによって起動させるためのコマンドです．

具体的には，ドライブ１，トラック０のセクタ１とセクタ２の内容をＰＣ－８０３１の５０００Ｈ～５１ＦＦＨ番地にロードして，５０００Ｈ番地へのジャンプを行ないます．

ユーザーが自ら組んだＰＣ－８０３１の制御プログラムをコマンドのみで動かす様な応用が考えられます．

パラメータはありません．

▷コマンド・コード１６

2.9.12　ディスク・ユニット中央のポート

ＰＣ－８０３１を御覧になった事の有る方は，すでにお気付きの事と思いますが，ＰＣ－８０３１の裏面には3個のコネクタが差し込める様に，３ヶ所の穴が並んでいます．

当然の事ですが，右端の穴はフロッピ・ディスク・ユニット用のインターフェース，例えばＰＣ－８０３３との接続ケーブルを接続するためのコネクタロであり，左端はドライブ３～４に相当する拡張ドライバをつなぐためのコネクタロになっています．それでは中央のコネクタロには何を接続するのでしょうか？

ＰＣ－８０３１のユーザーズ・マニュアルには，「未使用コネクタロ」などと示されていますが，実は中央のエッジ・カードには，プリンタをつなぐ事ができるのです．

ディスク・ユニットにプリンタなどをつないで何の役に立つのか疑問に思うユーザーもおられるかも知れませんが使い方によっては非常におもしろい利用方法が考えられます．

たとえばディスケット上のデータをプリンタから打ち出す場合には，一度コンピュータ本体にロードした上でプリンタに出力していました．ところが，ディスク・ユニットから直接プリンタへ出力が行なえれば，その間に本体の方には別の仕事をさせる事ができます．

又，プリンタへの出力中にディスク・ユニットへのアクセスが行なわれた場合には，一時プリンタへの出力を中断して，本体側との連絡を行なう事もソフト・ウェアによっては可能です．

このプリンタ・インターフェースは，ＰＣ－８０３１内のＣＰＵによって扱う事ができるのですが，残念ながらプリンタへ出力するためのプログラムは全くサポートされておりません．

そこでユーザー自らがプログラムを用意し，ＰＣ－８０３１内のＲＯＭを書き換えるか，ＲＡＭ上の空きエリアに転送した上で，ＰＣ－８０３１にコマンドを送って，プリンタ出力のためのサポート・プログラムにジャンプさせる事になります．

コネクタへのピン配列は，ＰＣ－８００１本体の配列と同じで，又，ＰＣ－８０３１のポートＦ６Ｈ～Ｆ７Ｈ番地がプリンタ制御用に利用できますので，ぜひこの機能を有効に利用してみてください．

なお，ポートＦ７Ｈの下位４ビットは，フロッピ・ディスク・コントローラの制御にも用いられています．

3
N-BASICおよびシステム・サブルーチン使用上の注意点

3.1　演算サブルーチンの利用方法

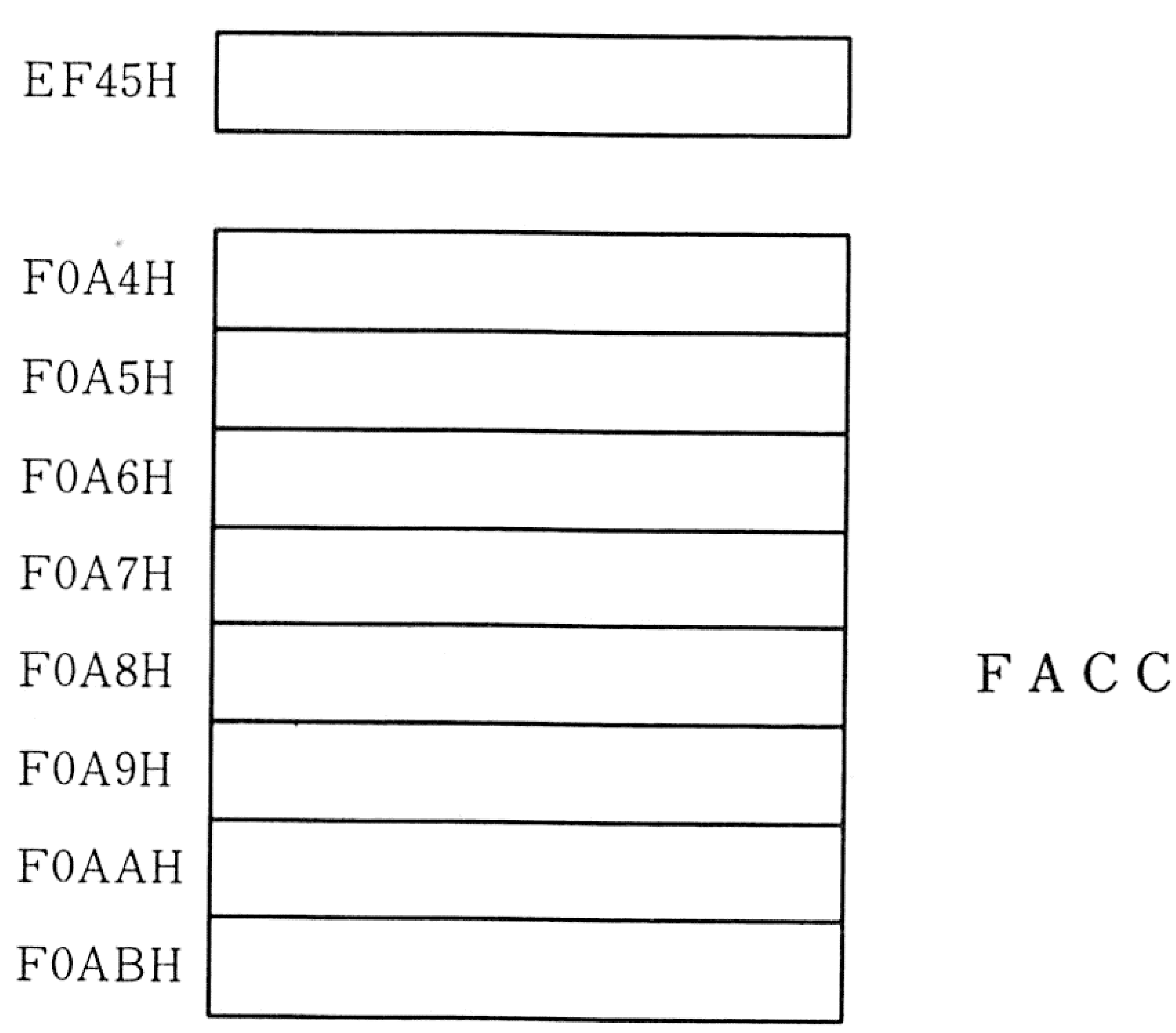
3.1.1　フローティング・アキュームレータ（FACC）

Ｚ－８０には浮動小数点の演算を行なうための命令など用意されていませんから，それらの演算の実行は全てソフト・ウェアによって行なわなければなりません．

Ｎ－ＢＡＳＩＣではフローティング・アキュームレータと呼ばれる仮想レジスタをメモリ上に取って浮動小数点の演算を行なっていますが，フローティング・アキュームレータへの数値の収納方法は，

整数型の数値
単精度実数型の数値
倍精度実数型の数値

のそれぞれの数値の型によって異なります．



3.1.1.1　整数型

整数型の場合には全て２バイトで収まりますから，Ｆ０Ａ８Ｈ番地に下位８ビットが，Ｆ０Ａ９Ｈ番地に上位８ビットが入っています．

ＥＦ４５Ｈ番地には，整数型を示す２が入ります．

EF45H	2

F0A4H	
F0A5H	
F0A6H	
F0A7H	
F0A8H	下位８ビット
F0A9H	上位８ビット
F0AAH	
F0ABH	

3.1.1.2　単精度実数型

単精度実数型の場合には，指数部１バイトと，仮数部３バイトに分けて，Ｆ０Ａ８Ｈ番地には仮数部の下位８ビットが，Ｆ０Ａ９Ｈ番地には仮数部の中位８ビットが，Ｆ０ＡＡＨ番地には仮数部の上位８ビットが入り，Ｆ０ＡＢＨ番地に指数部８ビットが入っています．

ＥＦ４５Ｈ番地には単精度実数型を示す４が入ります．

EF45H	4

F0A4H	
F0A5H	
F0A6H	
F0A7H	
F0A8H	仮数部下位８ビット
F0A9H	仮数部中位８ビット
F0AAH	仮数部上位８ビット
F0ABH	指数部８ビット

3.1.1.3　倍精度実数型

倍精度実数型の場合には仮数部７バイトと指数部１バイトを使い，仮数部の最下位がＦ０Ａ４Ｈ番地に，以下，Ｆ０ＡＡＨ番地まで仮数部が入り，Ｆ０ＡＢＨ番地には指数部１バイトが入っています．単精度型実数に仮数部４バイトを追加したと思えば良いでしょう．

ＥＦ４５Ｈ番地には倍精度実数型を示す８が入ります．

EF45H	8

F0A4H	仮数部最下位８ビット
F0A5H	
F0A6H	
F0A7H	
F0A8H	
F0A9H	
F0AAH	仮数部最上位８ビット
F0ABH	指数部８ビット

3.1.2　四則演算ルーチン

3.1.2.1　整数型数値の四則演算

整数型数値の四則演算は，ＥＤレジスタとＬＨレジスタとの間で行なわれ，演算結果はフローティング・アキュームレータに格納されます．

演算結果が単精度型実数として現れた場合には，フローティング・アキュームレータにも単精度型実数として格納され，ＥＦ４５Ｈ番地の内容も変化します．

整数型数値の四則演算サブルーチン

アドレス	演　　算
２８Ｄ２Ｈ	整数型数値の減算
２８ＤＤＨ	整数型数値の加算
２８ＦＤＨ	整数型数値の乗算
４ＢＥＡＨ	整数型数値の除算

3.1.2.2　単精度型実数の四則演算

単精度型実数の四則演算は，ＥＤＣＢレジスタ内の単精度実数と，フローティング・アキュームレータ内の単精度実数の間で行なわれ，演算結果はフローティング・アキュームレータに単精度型式で格納されます．

単精度型実数の四則演算サブルーチン

アドレス	演　　　算
２４０ＦＨ	単精度型実数の減算
２４１２Ｈ	単精度型実数の加算
２５４１Ｈ	単精度型実数の乗算
２５９ＣＨ	単精度型実数の除算

3.1.2.3　倍精度型実数の四則演算

倍精度実数型の四則演算は，フローティング・アキュームレータ内の倍精度実数型数値と，倍精度実数型数値同志の演算のためのサブ・フローティング・アキュームレータである，Ｆ０ＡＥＨ～Ｆ０Ｂ５Ｈ番地に渡る倍精度実数型数値の間で行なわれ，演算結果はフローティング・アキュームレータに倍精度実数型式で格納されます．

サブ・フローティング・アキュームレータでの表現方法はフローティング・アキュームレータのものと同じで，Ｆ０ＡＥＨ～Ｆ０Ｂ４Ｈ番地の７バイトに仮数部を，Ｆ０Ｂ５Ｈ番地に指数部を収納します．

倍精度型実数の四則演算サブルーチン

アドレス	演　　　算
２９Ｃ３Ｈ	倍精度型実数の減算
２９ＣＡＨ	倍精度型実数の加算
２ＡＦ４Ｈ	倍精度型実数の乗算
２Ｂ３７Ｈ	倍精度型実数の除算

3.1.3　関数処理ルーチンの使用方法

Ｎ－ＢＡＳＩＣに用意されているほとんどの数値関数は，単精度実数型式の数値のみをあつかい，入出力は全て，フローティング・アキュームレータである，Ｆ０Ａ８Ｈ～Ｆ０ＡＢＨ番地が用いられます．

実際に関数のシステム・サブルーチンを使用するには，フローティング・アキュームレータに単精度実数型式の数値データを，ＥＦ４５Ｈ番地に単精度実数型式を示す４を入れて，関数それぞれのエントリ・アドレスを呼び出すと，フローティング・アキュームレータの単精度実数に対して関数処理を行なった結果が，同じくフローティング・アキュームレータに単精度実数型式で格納されます．

アドレス	関　　　　数
２５０３Ｈ	ＬＯＧ（自然対数）
３１Ａ１Ｈ	ＳＱＲ（平方根）
３１Ｆ３Ｈ	ＥＸＰ（ｅのべき乗）
３２８３Ｈ	ＲＮＤ（乱数）
３２Ｆ６Ｈ	ＣＯＳ（コサイン）
３２ＦＣＨ	ＳＩＮ（サイン）
３３５ＤＨ	ＴＡＮ（タンジェント）
３３７２Ｈ	ＡＴＮ（アーク・タンジェント）

Ｎ－ＢＡＳＩＣの数値関数のなかでも一部の関数に限っては単精度実数型式の数値データのみでなく，整数型式や倍精度実数型式の数値をも，あつかう事ができます．

入出力にはフローティング・アキュームレータを使用しますが，関数処理サブルーチンを呼び出す前に，入力する数値データの型を示す数字をＥＦ４５Ｈ番地に入れておかなければなりません．

アドレス	関　　　　数
２６７１Ｈ	ＡＢＳ（絶対値）
２６８４Ｈ	ＳＧＮ（符号）
２７７ＦＨ	ＣＩＮＴ（整数型変換）
２７Ｂ３Ｈ	ＣＳＮＧ（単精度実数型変換）
２７ＤＦＨ	ＣＤＢＬ（倍精度実数型変換）
２８２ＣＨ	ＦＩＸ（整数部分）
２８３ＦＨ	ＩＮＴ（切り捨て）

3.1.4　マシン語による数値演算の例

ある程度のサブルーチンがわかっていても，実際にマシン語で演算を行なおうと思うと，それらのサブルーチンをどのように組み合わせれば良いのか迷ってしまう場合が多いものです．

そこで，システム・サブルーチンを十分に活用して，ある程度まとまった演算処理を行なうプログラム例を２本あげておきます．

3.1.4.1　例１　関数処理の実際

このプログラムは，平方根を求めるためのプログラムで，下記のＮ－ＢＡＳＩＣのプログラムと，全く同じ実行結果が得られるものです．

```
100 FOR I=0 TO 1000
110   PRINT "SQR(";I;") = ";SQR(I)
120 NEXT I
130 END
```

このように，０から１０００までの数値に対して平方根を求めるもので，演算結果はＣＲＴ画面に表示して行きます．

Ｎ－ＢＡＳＩＣとマシン語を組んだプログラムの実行例の一部を示します．Ｎ－ＢＡＳＩＣの実行例は，０から１０００までの数値を表示する部分で数字の両端にスペース（空白）が表示されていますが，これは，Ｎ－ＢＡＳＩＣが自動的に付け加えてしまうもので他意は無く，マシン語でも全く同じ表示を行なうようにしてあります．

さて，マシン語のプログラム例ですが，Ｎ－ＢＡＳＩＣ内のシステム・サブルーチンをできるだけ活用したので比較的簡単にまとめる事ができたかわりに，実行速度はそれほど上がっていません．プログラム自体はコメント等をながめる事で大体の見当は付くと思われますが，一応流れに添って解説します．

０～１０００の整数は，ＨＬレジスタ対に入れておき，インクリメント命令によって，プラス１を行なっていますが，これは正の整数に限定してあるために可能な事で，例えば－５００から５００まで，と言う事になれば正負を考慮した加算ルーチンを使用して，プラス１を行なわなければなりません．ループ処理からの脱出は，１６ビットの比較サブルーチンを呼んで，ＨＬとＤＥレジスタ対の値を比較して条件判断を行なっています．

又，ＨＬレジスタ対を利用した方法では，０～３２７６７の平方根しか求められません．それ以外は，負数とみなされるために平方根の演算サブルーチン内で，

```
Illegal function call
```

が出力されてしまいます．

ＳＱＲ（　　　）＝

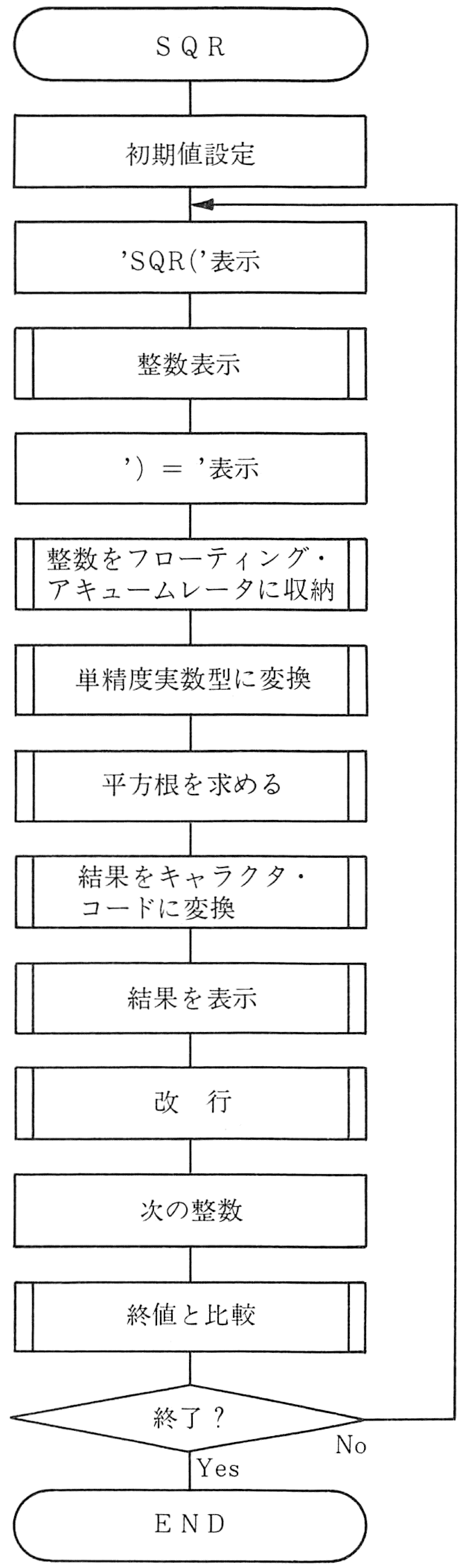
の表示には，５２ＥＤＨ番地の文字列出力サブルーチンを用い，カッコ内の数値はＨＬレジスタ対のデータを符号無１０進整数として出力する２Ｄ１３Ｈ番地を利用して表示します．

実際の演算は，ＨＬレジスタ対の値を，フローティング・アキュームレータに移して，単精度実数型に変換した上で，平方根を求めていますが，それらは全てシステム・サブルーチンに任せています．

演算結果の表示も，フローティング・アキュームレータの数値データをキャラクタ・コードの文字列に変換するサブルーチンを呼んだ後，文字列出力ルーチンによって行ないました．

```
                 ;*****************************
                 ;
                 ;        CALCULATE SQR
                 ;
                 ;*****************************
                 ;
                        ORG   9000H
                 ;
9000 210000             LD    HL,0          ; INITIAL NUMBER IN REGISTER HL
                 ;
9003 E5          LBL:   PUSH HL
9004 213890             LD    HL,MSG1       ; MESSAGE 1 IS 'SQR( '
9007 CDED52             CALL 52EDH          ; DISPLAY MESSAGE 1
900A E1                 POP   HL
                 ;
900B E5                 PUSH HL
900C CD132D             CALL 2D13H          ; DISPLAY REGISTER HL DECIMAL
900F 213E90             LD    HL,MSG2       ; MESSAGE 2 IS ' ) = '
9012 CDED52             CALL 52EDH          ; DISPLAY MESSAGE 2
9015 E1                 POP   HL
                 ;
9016 E5                 PUSH HL
9017 CD9C27             CALL 279CH          ; STORE HL IN FLOWTING ACCUMULATOR
901A CDB327             CALL 27B3H          ; CONVERT INTO SINGLE PRECISION
901D CDA131             CALL 31A1H          ; CALCULATE SQR
9020 CD222D             CALL 2D22H          ; CONVERT INTO STRING
9023 CDED52             CALL 52EDH          ; DISPLAY RESULT
9026 CDCA5F             CALL 5FCAH          ; DISPLAY CR CODE AND LF CODE
9029 E1                 POP   HL
                 ;
902A 23                 INC   HL            ; INCREMENT NUMBER
902B 11E903             LD    DE,1000+1     ; ENDING NUMBER+1 IN REGISTER DE
                 ;
902E E5                 PUSH HL
902F CDD35E             CALL 5ED3H          ; COMPARE REGISTER PAIRS
9032 E1                 POP   HL
                 ;
9033 20CE               JR    NZ,LBL        ; IF DE<>HL THEN CALCULATE AGAIN
9035 C3665C             JP    5C66H         ; JUMP MONITOR
                 ;
9038 53515228 MSG1:     DB    'SQR( ',0     ; MESSAGE 1
903C 2000
903E 2029203D MSG2:     DB    ' ) = ',0     ; MESSAGE 2
9042 2000
                 ;
```

```
SQR( 61 ) = 7.81025
SQR( 62 ) = 7.87401
SQR( 63 ) = 7.93726
SQR( 64 ) = 8
SQR( 65 ) = 8.06226
SQR( 66 ) = 8.12404
SQR( 67 ) = 8.18536
SQR( 68 ) = 8.24621
SQR( 69 ) = 8.30663
SQR( 70 ) = 8.3666
SQR( 71 ) = 8.42615
SQR( 72 ) = 8.48528
SQR( 73 ) = 8.54401
SQR( 74 ) = 8.60232
SQR( 75 ) = 8.66026
SQR( 76 ) = 8.7178
SQR( 77 ) = 8.77496
SQR( 78 ) = 8.83176
SQR( 79 ) = 8.8882
SQR( 80 ) = 8.94428
```



SQR
初期値設定
'SQR('表示
整数表示
')＝'表示
整数をフローティング・アキュームレータに収納
単精度実数型に変換
平方根を求める
結果をキャラクタ・コードに変換
結果を表示
改　行
次の整数
終値と比較
終了？
No
Yes
END

3.1.4.2　例2 数値データの入出力

このプログラムは，円の半径を入力して，面積を求めるためのプログラムで，下記のＮ－ＢＡＳＩＣのプログラムと全く同じ実行結果が得られるものです．

```
100 INPUT "R ";R$
110 PRINT "R = ";R$
120 PRINT "S =";
130 R=VAL(R$)
140 PRINT R*R*3.14159
150 PRINT
160 GOTO 100
```

プログラム実行中の画面コピーを行なったものを示します．

このように，半径Ｒを入力すると，

Ｒ＊Ｒ＊３．１４１５９（πr^2）

によって求められた円の面積をＣＲＴ画面に表示します．なお，プログラムは無限ループになっており，実行を止めるにはリセット・スイッチを押さなければなりません．

このプログラムでは，実行速度がＮ－ＢＡＳＩＣのものより，かなり向上しています．

特に，半径Ｒの値として，

０．０００００００００１

などを入力した場合に，両者の速度の差が顕著にあらわれ，Ｎ－ＢＡＳＩＣの方ではあきらかに待ち時間があるのに対して，マシン語のプログラムではほとんど瞬間的に演算結果が表示されます．

さて，プログラムですが最初に，文字列出力サブルーチンを利用してプロンプトを表示した後，３Ｅ５ＣＨ番地のサブルーチンを呼んで，キャラクタ・コードでインプットを行ない，２ＢＢＥＨ番地のサブルーチンによって数値への変換とフローティング・アキュームレータへの収納を行ないます．

つまり，フローティング・アキュームレータへの数値インプットの定石は，

```
CALL  3E5CH     ;入力
INC   HL
CALL  2BBEH     ;変換
```

と，なります．

次に，入力された半径Rを２乗しますが，これは，フローティング・アキュームレータのRを単精度実数型に変換するサブルーチンを呼んだ後，ＥＤＣＢレジスタにコピーして，単精度実数型の乗算サブルーチンによって，かけ合わせています．この時に注意を要するのが，Ｚ８０の特性と同じようにメモリ上の単精度実数型数値４バイトをレジスダ上では逆の順序で収納する点です．コピー時にはシステム・サブルーチンを利用すれば良いのですが，定数を代入する場合には良く注意してください．

▷メモリ

F0A8H	←	仮数部下位８ビット
F0A9H	←	仮数部中位８ビット
F0AAH	←	仮数部上位８ビット
F0ABH	←	指数部　８ビット

▷レジスタ

E	←	仮数部下位８ビット
D	←	仮数部中位８ビット
C	←	仮数部上位８ビット
B	←	指数部　８ビット

半径Rの２乗に円周率をかける部分では，ＥＤＣＢレジスタに，３．１４１５９を４バイトの単精度実数型として表わしたものを入れて先程と同じ乗算サブルーチンを呼んでいます．
円周率は次のように表現できます．

▷３．１４１５９（円周率）

ＤＢＨ　：　仮数部下位８ビット
０ＦＨ　：　仮数部中位８ビット
４９Ｈ　：　仮数部上位８ビット
８２Ｈ　：　指数部　　８ビット

最後に，フローティング・アキュームレータに求めた面積を表示しますが，これは，２Ｄ２２Ｈ番地のキャラクタ・コードの文字列に変換するためのサブルーチンによって，文字列に変換した後，通常の文字列出力サブルーチンで表示しています．

一般的に，フローティング・アキュームレータ内の数値データを出力するためには

```
CALL  2D22H   ; 変換
CALL  52EDH   ; 出力
```

を実行します．

全ての処理を終了した後は，プログラムの先頭にジャンプして，再度半径Ｒの入力をくり返します．

```
R = 0
S = 0

R ? 1
R = 1
S = 3.14159

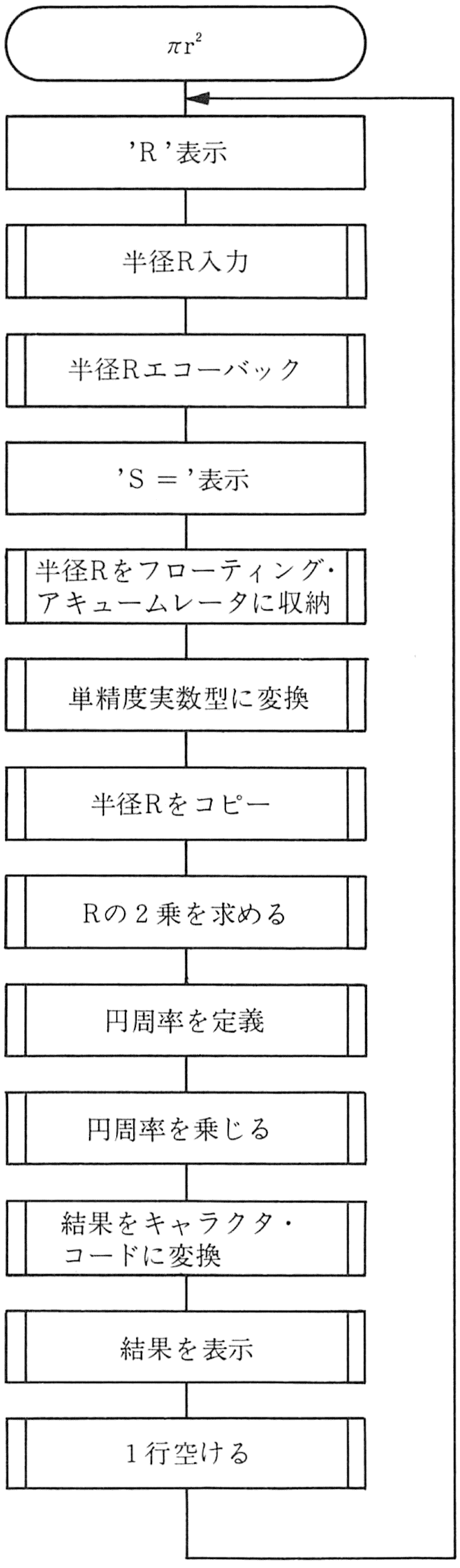
R ? 2
R = 2
S = 12.5664

R ? 3
R = 3
S = 28.2743

R ? 5000
R = 5000
S = 7.85398E+07

R ? 0.1
R = 0.1
S = .0314159
```

```
               ;*****************************
               ;
               ;   CALCULATE AREA OF CIRCLE
               ;
               ;*****************************
               ;
                      ORG  09000H
               ;
9000 214690    LBL:   LD   HL,MSG1        ; MESSAGE 1 ´R ´
9003 CDED52           CALL 52EDH          ; DISPLAY MESSAGE 1
9006 CD5C3E           CALL 3E5CH          ; INPUT R FROM KEYBOARD
9009 23               INC  HL             ; START ADDRESS
               ;
900A E5               PUSH HL
900B 214990           LD   HL,MSG2        ; MESSAGE 2 ´R = ´
900E CDED52           CALL 52EDH          ; DISPLAY MESSAGE 2
9011 E1               POP  HL
               ;
9012 E5               PUSH HL
9013 CDED52           CALL 52EDH          ; DISPLAY R
9016 CDCA5F           CALL 5FCAH          ; DISPLAY CR CODE AND LF CODE
9019 214E90           LD   HL,MSG3        ; MESSAGE 3 ´S =´
901C CDED52           CALL 52EDH          ; DISPLAY MESSAGE 3
901F E1               POP  HL
               ;
9020 CDBE2B           CALL 2BBEH          ; STORE R IN FLOWTING ACCUMULATOR
9023 CDB327           CALL 27B3H          ; CONVERT INTO SINGLE
9026 CDBD26           CALL 26BDH          ; MOVE R TO REGISTER EDBC
9029 CD4125           CALL 2541H          ; CALCULATE FACC*EDBC IN FACC
               ;
902C 1EDB             LD   E,0DBH         ;
902E 160F             LD   D,0FH          ; PAI 3.14159
9030 0E49             LD   C,49H          ;    IN REGISTER EDCB
9032 0682             LD   B,82H          ;
               ;
9034 CD4125           CALL 2541H          ; CALCULATE FACC*EDCB IN FACC
               ;
9037 CD222D           CALL 2D22H          ; CONVERT INTO STRING
903A CDED52           CALL 52EDH          ; DISPLAY S
903D CDCA5F           CALL 5FCAH          ; DISPLAY CR CODE AND LF CODE
9040 CDCA5F           CALL 5FCAH          ; DISPLAY CR CODE AND LF CODE
               ;
9043 C30090           JP   LBL            ; JUMP ORIGIN
               ;
9046 522000    MSG1:  DB   ´R ´,0         ; MESSAGE 1
9049 52203D20  MSG2:  DB   ´R = ´,0       ; MESSAGE 2
904D 00
904E 53203D00  MSG3:  DB   ´S =´,0        ; MESSAGE 3
               ;
```


πr²
'R'表示
半径R入力
半径Rエコーバック
'S ='表示
半径Rをフローティング・アキュームレータに収納
単精度実数型に変換
半径Rをコピー
Rの2乗を求める
円周率を定義
円周率を乗じる
結果をキャラクタ・コードに変換
結果を表示
1行空ける

3.2 N－BASICコマンド処理ルーチンの利用方法

N－BASICインタプリタは，HLレジスタ対をテキスト・ポインタとしてN－BASICのプログラムを解析し，一つの命令や関数などを発見した時点で，それぞれの処理ルーチンをサブルーチンとしてコールします．

ゆえに，これらの処理ルーチンをマシン語のプログラム中から呼び出す事によって，N－BASICで行なう事のできるほとんどの機能を利用する事が可能です．

正攻法では無いかもしれませんが，いくつかの例をあげておきましょう．

なお、それぞれの処理ルーチンの先頭アドレスは本書付属のコマンド⟷ジャンプ・テーブルの対応表を参照してください．又，処理ルーチン内では全てのレジスタの値が保障されないと考えた方が良く，エラーなどが発生すれば，容赦なくN－BASICの制御ルーチン内へ飛び込んでしまいます．十分に御注意ください．

3.2.1 ドットのセットおよびリセット

まず，N－BASICのドット・グラフィクによって，CRT画面へのドットのセットとリセットを行なってみましょう．

N－BASICでは，PSET，PRESETを見つけた時点で，それぞれ次のアドレスをコールします．

PSET　　⟶　06B8H
PRESET　⟶　0705H

コールする時点で，HLレジスタ対には入力パラメータの先頭アドレスを指しており，入力パラメータは，N－BASICのプログラム入力時と同じ様にキャラクタ・コードの文字列で格納されています．

実際に，この機能を利用して，座標（10，10）にドットをセットするサブルーチン例を示します．

```
        LD      HL, LBL
        CALL    06B8H
        RET
LBL:DB      '(10, 10)'
        DB      0
```

同じ位置のドットをリセットするためには次の様なサブルーチンを用います.

```
        LD      HL, LBL
        CALL    0705H
        RET
LBL:DB      '(10, 10)'
        DB      0
```

パラメータの最後は0で終えてあります.

N－BASICの命令に対応した処理ルーチンの中で，確実に動作するものをあげておきます．他にも多くのルーチンが利用できるはずです.

なお，4877H番地のLINEルーチンは，カラー・モードの一行単位の機能指定に使用するもので，ラインを引くものでありません.

```
PSET     →  06B8H
PRESET   →  0705H
LOCATE   →  0792H
WIDTH    →  0843H
CONSOLE  →  0884H
COLOR    →  0951H
LINE     →  4877H
```

etc ....

3.3　システム・サブルーチンの実行速度

プログラムの実行中，頻繁にキーボードをスキャンし，入力に応じて作業を変更する場合などには，N－ＢＡＳＩＣ内部のキー・スキャン・ルーチンを多用する事は多少問題があります．

なぜならば，ＲＯＭ内部のキー・スキャン・ルーチンは，キー入力の有無にかかわらずデータ・テーブルにしたがってキャラクタ・コードを求めているため，キー入力の無い場合にも，恐しく時間がかかってしまうからです．

これは，実行速度が問題となる，システム・プログラム言語，ワード・プロセッサ，制御プログラムなどでは特に注意が必要で，せっかくマシン語で組んだプログラムが実用にならない場合さえ有り得ます．

このような，実行速度上の問題点は，システム・サブルーチンを活用する場合には有りがちな事で，これらのサブルーチンが，ある程度汎用を目指して組まれている事を考えれば仕方の無い事でしょう．特に画面制御関係のサブルーチンは実行速度に問題が多いようです．

参考のために，上記したキー・スキャン・ルーチンをユーザーのプログラムでサポートしたサブルーチン例を示します．

このサブルーチン例では，０ＦＡＣＨ番地からのシステム・サブルーチンと全く同じ動作を行ないますが，キー入力の無い場合には，ほとんど時間を浪費せずにもどって来ます．

```
                  ;*****************************
                  ;
                  ; KEYBOARD SCANNING SUBROUTINE
                  ;
                  ;*****************************
                  ;
                          ORG   9000H
                  ;
9000 C5                   PUSH BC
9001 0E00                 LD    C,0
                  ;
9003 ED40         LOOP:   IN    B,(C)
9005 79                   LD    A,C
9006 FE08                 CP    8
9008 78                   LD    A,B
9009 2002                 JR    NZ,NOTCTL
                  ;
900B F6F0                 OR    0F0H
                  ;
900D 3C           NOTCTL:INC    A
900E 2009                 JR    NZ,INPUT
                  ;
9010 0C                   INC   C
9011 79                   LD    A,C
9012 D60A                 SUB   10
9014 20ED                 JR    NZ,LOOP
                  ;
9016 37                   SCF
9017 C1                   POP   BC
9018 C9                   RET
                  ;
9019 CDAC0F       INPUT:  CALL  0FACH
901C C1                   POP   BC
901D C9                   RET
                  ;
```

このような，ちょっとした工夫によって，実用にならないと思われていたプログラムやサブルーチンが実用的に使用できるようになった事例も数多く存在します．ハードウェアやプログラムの限界ではないかと疑がう前に，いろいろと試してみる事を勧めます．

3.4　ガベージ・コレクション

マイクロ・ソフト系のＢＡＳＩＣ特有の欠点としてガベージ・コレクションをあげることができます．

インタプリタは文字列の演算や文字変数への代入が必要になった場合には，ストリング・エリアを文字列を退避しておくためのスタックとして用いますが，その際にストリング・エリア内を整理しながら，不用になった部分を前につめて使用したのでは実行速度の点で問題があります．そのため，普通はストリング・エリア内に多少の無駄が有っても無視をして使って行き，ストリング・エリアが一杯になった時点で，不用な部分をさがし出してストリング・エリア内の整理をし直します．

このストリング・エリアの整理には下手をすると５分近くかかる事もあり，その間，当然の事ですがプログラムの実行は停止してしまいますから文字列処理が中心のプログラムや実行速度が生命である制御プログラムでは大問題となります．

ガベージ・コレクションはマイクロ，ソフト系ＢＡＳＩＣの宿命ですが，プログラムの工夫によって，最少限度に押える事は可能です．以下に，ガベージ・コレクションを防ぐためのテクニックをいくつか紹介しますが，いずれも，ストリング・エリアを一杯にしない事を最も重視しています．

まず，Ｎ－ＢＡＳＩＣのＣＬＥＡＲによってできる限り多くのストリング・エリアを確保します．プログラム自身を格納できる最少限度のエリアと配列変数，単純変数のためのエリアを残したほとんど全てのフリー・エリアをストリング・エリアとして確保しておくのが良いでしょう．

また，Ｎ－ＢＡＳＩＣの関数である，ＶＡＲＰＴＲを活用して，文字変数のポインタ（ストリング・デュプリケータ）を，直接操作し，インタプリタへの負担をできる限り，プログラマのプログラミング・テクニックによって，おぎなう事も有効です．単にＢＡＳＩＣによるプログラミングであっても，インタプリタの構造を熟知したものと，そうでないものの間には，歴然とした差があらわれるものです．

以下に２種類のサブルーチン例を示します．両者共に，文字型の配列変数Ａ＄（０）～Ａ＄（９９８）に格納されている文字型のデータ群を，同じくＡ＄（１）～Ａ＄（９９９）に移し換えるサブルーチンです．このように，配列変数内のデータ群をシフトする機会は，特に業務処理などには多発しますが，前者は通常使用される命令のみによって普通に組んだものであり，後者はポインタを直接操作する事によって，ストリング・エリアの使用を最少限に押えたものです．

プログラムの解り易さや実行速度から見れば，圧倒的に前者の方が有利ですが，実際のプログラム中では後者の手法を用いなければならない場合も数多く存在します．

```
100 '********************************
110 '
120 '  SHIFT DIMENTION SUBROUTINE 1
130 '
140 '********************************
150 '
160 FOR I=1 TO 999
170   A$(I)=A$(I-1)
180 NEXT I
190 RETURN
```

```
100  '********************************
110  '
120  '  SHIFT DIMENTION SUBROUTINE 2
130  '
140  '********************************
150  '
160 FOR I=VARPTR(A$(999))+2 TO VARPTR(A$(1)) STEP -1
170   POKE I,PEEK(I-3)
180 NEXT
190 RETURN
```

3.5　64KバイトRAMのシステム

N－BASICのような非常にすぐれた言語であっても，長い間使っているうちには，ユーザーなりの不満な点があらわれるものです．通常N－BASICはROMに焼き込まれているため，ユーザーが勝手に内容を変更する事はできませんが，PC－8001を64KバイトRAMのシステムとして，N－BASICをRAM上に移してしまえば，N－BASICを自由にパッチして自分なりのシステムをつくり上げる事もできますし，オールRAMのシステムではCP／Mなどを利用する事も可能です．

PC－8001を64KバイトRAMのシステムとして利用するためには普通NECのPC－8011，PC－8013またはPC－8012＋PC－8012－02を組み合わせますが，最近では，手軽にオールRAMのシステムを利用するために，秀和システム・トレーディング（株）より「ADC－32KR」というRAMユニットが発売されています．これは，ミニ・フロッピ・ディスクを利用するためのI／Oポートも含んでおり，64KバイトRAMのディスク・システムを低価格で実現できるものです．

参考のために，N－BASICおよび拡張ROMの内容を，RAM上に移して制御を移

すプログラム例をあげておきます．

この例では，ＰＣ－８０１２上バンクゼロのＲＡＭボードを指定しています．

```
                    ;********************
                    ;
                    ;   N-BASIC ON RAM
                    ;
                    ;********************
                    ;
                            ORG   0EE00H
                    ;
EE00 3E10                   LD    A,10H
EE02 D3E2                   OUT   (0E2H),A
                    ;
EE04 010080                 LD    BC,8000H
EE07 110000                 LD    DE,0
EE0A 210000                 LD    HL,0
EE0D EDB0                   LDIR
                    ;
EE0F 3E11                   LD    A,11H
EE11 D3E2                   OUT   (0E2H),A
                    ;
EE13 C38100                 JP    81H
                    ;
```

また，この様なシステムでは，拡張ＲＯＭの領域をＲＡＭ化して，Ｎ－ＢＡＳＩＣのテキスト・エリアを８Ｋバイト増加する事もできます．

具体的には，８０００Ｈ番地からの割り込みテーブルを６０００Ｈ番地に移してインタラプト・ベクタ・レジスタに６０Ｈをセットし，ワーク・エリアも変更する必要がありますが，次の様なプログラムによって，これら一連の作業を行なう事ができます．

```
                    ;********************
                    ;
                    ;   TEXT TOP 6000H
                    ;
                    ;********************
                    ;
                            ORG   0EE00H
                    ;
EE00 210060                 LD    HL,6000H
EE03 C3E917                 JP    17E9H
                    ;
```

プログラム実行後は，Ｎ－ＢＡＳＩＣのメッセージを出力し，コールド・スタートが行なわれます．

3.6 Ｎ－ＢＡＳＩＣのバージョン

3.6.1 バージョン１.０と１.１の相違点

Ｎ－ＢＡＳＩＣは，ＮＥＣのパーソナル・コンピュータＰＣ－８００１のために開発されたベーシック・インタプリタですが，当初のＮ－ＢＡＳＩＣには，わずかながら不完全な部分がありました．

そこで，当初のＮ－ＢＡＳＩＣに多少の修正をおこなった新しいバージョンのＲＯＭがつくられ，シリアル・ナンバの頭文字が「Ｐ」からのＰＣ－８００１には，新しいタイプのＲＯＭが標準実装されています．

Ｎ－ＢＡＳＩＣが当初のものか，新しいバージョンのものであるかは，コールド・スタート時のメッセージが

```
NEC PC-8001 BASIC Ver 1.0
Copyright 1979 (C) by Microsoft

Ok
```

と表示されるか，

```
NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

Ok
```

と表示されるかによって判別でき，新しいタイプのROMは販売されているため，旧タイプのN－BASICから新しいバージョンのN－BASICへは，2個のICを交換する事で簡単に行なう事ができます.

64Kバイト RAMのシステムを用いている場合などには，新しいバージョンへの変更点，数10バイトのみを変更して利用すれば，わざわざROMを交換する必要もないと思います.

N－BASICの主な変更点としては，

▷スクリーン・エディタのデバッグ

▷コールド・スタートのデバッグ

▷ターミナル・モードの改良

の3ヶ所をあげる事ができます.

スクリーン・エディタの1文字欠落は有名なバグで，プログラムなどの入力時に，CRT画面で2行以上に渡る入力データの改行直前のキャラクタが欠落してしまうものです.

プログラム入力時には，リターン・キーを注意して押し直す事によって防ぐ事ができますが，プログラム実行中のインプットにはかなり苦しんだ方も多いと思われます．特に40字モードの場合は多発し，PC－8001に付属されているデモンストレーション・テ

ープのメイリング・リストなども正常に実行できなかったようです.

バージョン1．1のN－BASICでは，この1文字欠落がほとんど起こらなくなっています．それでも条件によっては，ごくまれですが末尾の1文字が欠落します.

N－BASICでは，6000H～7FFFH番地がROM用の空きソケットとして残してありますが，この6000H番地に41Hを，6001番地に42Hを焼いておくと，コールド・スタート時に6002H番地をサブルーチンとしてコールする様になりました。これによって，リセットと同時にユーザー自作のシステムやモニタなどを起動させる事が可能となり，増設ROM領域をより有効に利用する事ができます.

ターミナル・モードは，N－BASICのTERMによって起動されるモードで，RS－232Cを通した交信によって，PC－8001をターミナルとして利用しようというものです.

旧バージョンのN－BASICでは，ターミナル・モードにおいても全てのキー入力に対してオート・リピートが働らいたのに対して，新しいN－BASICでは，ターミナル・モード時のストップ・キーに対してのみ，リピート機能が無効になります.

3.6.2 ＰＣ－８８０１のＮ－ＢＡＳＩＣ

ＰＣ－８８０１には，さらにバージョン・アップを行なったバージョン１．２のＮ－ＢＡＳＩＣが標準実装されていますが，これらのＮ－ＢＡＳＩＣの間では，テキスト・レベルにおいては完全なコンパチビリティが得られ，マシン・ランゲージにおいてもほとんどのプログラムがそのまま実行できます．

```
NEC PC-8001 BASIC Ver 1.2
Copyright 1979 (C) by Microsoft

Ok
```

バージョン１．２への主な変更点は，

▷キー入力ルーチンの変更

▷タイマ・イニシアライズの削除

▷ＣＲＴコントローラ制御の変更

の３点です．

ＰＣ－８８０１では，キャピタル・ロック・キーが加えられるなど，キーボードに大幅な改良がみられます．この改良に対応するためＮ－ＢＡＳＩＣバージョン１．２のキー・スキャニング・ルーチンには数ヶ所の変更点がみられます．

また，ＰＣ－８８０１のタイマは，バッテリによってバック・アップが行なえますが，

せっかくの機能もＮ－ＢＡＳＩＣがホット・スタートするたびにタイマのリセットが行なっては意味がありません.

このため，ＰＣ－８８０１のＮ－ＢＡＳＩＣでは一切タイマのイニシアライズを行ないません.

Ｎ８８－ＢＡＳＩＣでは，６４０×４００ドットのフル・グラフィックをサポートしていますが，従来のＣＲＴディスプレイでは，この解像度に対応できないため，水平周波数２４．８KHzの専用高解像度のＣＲＴディスプレイも接続できるように考えられています.

どちらのＣＲＴデイスプレイを接続するのかは，ＰＣ－８８０１裏面のジャンパ・スイッチによって切り換えますが，両者によってＣＲＴコントローラのイニシアライズ時に指定するスクリーン・フォーマットの指定値が異なります.

このため，ＷＩＤＴＨの変更時等にコールされるＣＲＴコントローラのイニシアライズ・ルーチンが変更されています.

3.6.3 N－BASIC Ver1.0→1.1→1.2 対照表

```
0F0B: F1 -> E6
0F0C: 0C -> 5F
0F15: F1 -> E6
0F16: 0C -> 5F
17A7: 23 -> 20
17A8: 20 -> 06
17A9: 05 -> CD
17AA: FE -> ED
17AB: 42 -> 5F
1850: 30 -> 31
1C5E: 3A -> CD
1C5F: B6 -> D8
1C60: ED -> 5F
1C61: 3D -> 00
5FE6: FF -> AF
5FE7: FF -> 32
5FE8: FF -> 75
5FE9: FF -> EA
5FEA: FF -> C3
5FEB: FF -> F1
5FEC: FF -> 0C
5FED: FF -> 23
5FEE: FF -> 7E
5FEF: FF -> FE
5FF0: 00 -> 42
5FF1: 39 -> C9
```

```
09F9: 3A -> C3
09FA: 67 -> 74
09FB: EA -> 7F
0FD9: 41 -> 07
0FDA: 10 -> 7F
0FEC: A1 -> 25
0FED: 10 -> 7F
1624: C0 -> C9
164D: 04 -> 82
164E: CD -> 32
164F: 75 -> 7B
1650: 16 -> EA
1651: 01 -> C9
1710: 3B -> AE
1711: 17 -> 7F
1757: 3A -> AF
1758: 00 -> D3
1759: 00 -> E7
175A: 3A -> D3
175B: 00 -> F8
175C: 40 -> 00
179F: CF -> 00
17A0: 0A -> 7F
17F9: C9 -> A5
17FA: 00 -> 7F
1850: 31 -> 32
187E: C9 -> A5
187F: 00 -> 7F
1880: C9 -> A5
1881: 00 -> 7F
```

4
ユーティリティ・プログラム

4.1 N－BASIC　ROM　4バイト・サーチ

このプログラムは，N－BASICのROMエリアである，００００H～５FFFH番地のなかから任意のデータが収納されているアドレスをさがし出すためのもので，本書の執筆中にも何度か利用した大へん利用価値の高いユーティリティ・プログラムです．メイン・ルーチンはZ８０マシン語で，N－BASICのデータとなっていますが，入出力はN－BASICが管理していますから，Z８０の機種であれば，簡単に移植できる事と思います．

4.1.1　使用方法

サーチするためのキー・ワードは，１６進数とキャラクタ・コードの両方を用いる事ができますが，キャラクタ・コードでサーチする場合には，キー・ワードの末尾に，ピリオド「．」を付けて区別します．

たとえば，Z８０の無条件絶対ジャンプ命令であるC３Hをキー・ワードとして，全アドレスをリスト・アップしたい場合には，

```
KEY WORD 1 ? C3↵
KEY WORD 2 ? ↵
```

の下線部分を入力し，「NEC」の文字列でサーチする場合には，

```
KEY WORD 1 ? N.↵
KEY WORD 2 ? E.↵
KEY WORD 3 ? C.↵
KEY WORD 4 ? ↵
```

のように，キャラクタの末尾に，ピリオド「．」をつけて入力します．

以上のように，４バイトまでのキー・ワードでサーチを行ないますが，サーチしたアドレスの合計が１２８個以上になった場合には先頭の１２７個を出力後「＞＞＞＞　Over　１２８！」のメッセージを出し，キー・ワードを発見できなかった場合にも，「Not found！」を表示します．

```
100 '******************************
110 '
120 '   N-BASIC ROM 4 BYTE SEARCH
130 '
140 '******************************
150 '
160 CLEAR 300,&H97FF
170 DEFUSR0=&H9800
180 DEFINT A-Z
190 FOR I=&H9800 TO &H983F
200   READ D$
210   POKE I,VAL("&H"+D$)
220 NEXT
230 BY=0
240 PRINT"KEY WORD";BY+1;"? ";:LINE INPUT I$
250 IF I$="" THEN 290 'return key only then exit from input
260 IF RIGHT$(I$,1)="." THEN I$=HEX$(ASC(I$)) 'with . then ascii else hex
270 POKE &HA001+BY,VAL("&H"+I$)
280 BY=BY+1:IF BY<4 THEN 240 'continue input
290 IF BY=0 THEN END 'none key word then end
300 POKE &HA000,BY
310 A=USR0(0)
320 B=PEEK(&HA005)
330 PRINT
340 IF B=0 THEN PRINT"Not found!":GOTO 420
350 FOR I=&HA006 TO &HA004+B*2 STEP 2
360   PRINT RIGHT$("00"+HEX$(PEEK(I+1)),2);
370   PRINT RIGHT$("00"+HEX$(PEEK(I  )),2);
380   PRINT " ";
390 NEXT
400 IF B=&HFF THEN PRINT:PRINT">>>> Over 128!";
410 PRINT
420 PRINT
430 GOTO 230 'search again
440 '
450 DATA DD,21,06,A0 :' 9800: DD2106A0 LD   IX,A006H
460 DATA AF          :' 9804: AF       XOR  A
470 DATA 32,05,A0    :' 9805: 3205A0   LD   (A005H),A
480 DATA 21,00,00    :' 9808: 210000   LD   HL,0000H
490 DATA 01,00,60    :' 980B: 010060   LD   BC,6000H
500 DATA 11,01,A0    :' 980E: 1101A0   LD   DE,A001H
510 DATA 1A          :' 9811: 1A       LD   A,(DE)
520 DATA ED,B1       :' 9812: EDB1     CPIR
530 DATA E0          :' 9814: E0       RET  PO
540 DATA C5          :' 9815: C5       PUSH BC
550 DATA 2B          :' 9816: 2B       DEC  HL
560 DATA E5          :' 9817: E5       PUSH HL
570 DATA 3A,00,A0    :' 9818: 3A00A0   LD   A,(A000H)
580 DATA 47          :' 981B: 47       LD   B,A
590 DATA 05          :' 981C: 05       DEC  B
600 DATA 28,08       :' 981D: 2808     JR   Z,9827H
610 DATA 23          :' 981F: 23       INC  HL
620 DATA 13          :' 9820: 13       INC  DE
630 DATA 1A          :' 9821: 1A       LD   A,(DE)
640 DATA BE          :' 9822: BE       CP   (HL)
650 DATA 20,13       :' 9823: 2013     JR   NZ,9838H
660 DATA 18,F5       :' 9825: 18F5     JR   981CH
670 '
680 DATA E1          :' 9827: E1       POP  HL
690 DATA DD,75,00    :' 9828: DD7500   LD   (IX+0),L
700 DATA DD,23       :' 982B: DD23     INC  IX
710 DATA DD,74,00    :' 982D: DD7400   LD   (IX+0),H
720 DATA DD,23       :' 9830: DD23     INC  IX
730 DATA E5          :' 9832: E5       PUSH HL
740 DATA 21,05,A0    :' 9833: 2105A0   LD   HL,A005H
750 DATA 34          :' 9836: 34       INC  (HL)
760 DATA 7E          :' 9837: 7E       LD   A,(HL)
770 DATA E1          :' 9838: E1       POP  HL
780 DATA 23          :' 9839: 23       INC  HL
790 DATA C1          :' 983A: C1       POP  BC
800 DATA FE,FF       :' 983B: FEFF     CP   FFH
810 DATA C8          :' 983D: C8       RET  Z
820 DATA 18,CE       :' 983E: 18CE     JR   980EH
```

```
run
KEY WORD 1 ? C3
KEY WORD 2 ?

0004 0008 000B 0010 0013 0018 001D 0020 0023 0028 002B 0030 0035 0038 007E 0096
019D 01E6 02AD 0366 0477 04AC 0642 06E7 06EC 0773 0784 0881 09F9 0AE6 0AF6 0C19
0C96 0CD4 0CE9 0D31 0D5D 0DA8 0DB5 0F19 0F72 12D6 14A7 159C 15C3 15FD 178D 181D
182F 1877 188E 18E9 191A 1A2D 1A4D 1A68 1A70 1A74 1AB3 1E6D 1F9F 1FB3 1FC6 20E9
2127 212D 21ED 21F6 224C 225F 22D2 22DB 22DE 2361 236C 2373 237C 23A6 23B2 23ED
24A1 24AE 25FD 25FE 2635 264C 266E 268D 27CD 27DC 27F7 27FE 283C 28A8 28FA 2927
293F 294D 2984 299A 29AF 2A1F 2A46 2B1B 2B22 2B87 2B88 2BB4 2C45 2CB8 2CF5 2D08
2D1F 2E66 2EBB 2F07 2F65 3196 3230 32E0 333D 336F 34FF 3595 3664 36FF 37D5 3B29
3B2C 3BC3 3BD6 3C13 3C42 3CD1 3CE9 3D73 3E62 3EE5 3EF0 3F44 3F80 3FAC 3FCC 3FCF
3FE2 4002 4035 4055 4061 407F 40A3 40E7 427D 42A3 42F0 42FF 432C 4369 4380 4452
44A7 453A 4569 45A0 466D 46C1 46CC 46FF 4726 473F 47C6 47FC 4812 4862 48A9 49A0
49D4 4A05 4A70 4A88 4B29 4BF6 4C27 4DB4 4DD5 4EF0 4FBF 5071 507F 50EF 519E 525F
526D 52E9 5397 540E 5445 54B5 5526 56AA 56B0 576B 57D8 57DE 57F0 5849 5866 58C7
58D6 5915 5943 5A98 5ADF 5AE8 5B3C 5B7D 5BBF 5BED 5C29 5D0F 5DAB 5DDB 5DE3 5E0F
5E80 5F9B 5FD1 5FEA

KEY WORD 1 ? N.
KEY WORD 2 ? E.
KEY WORD 3 ? C
KEY WORD 4 ?

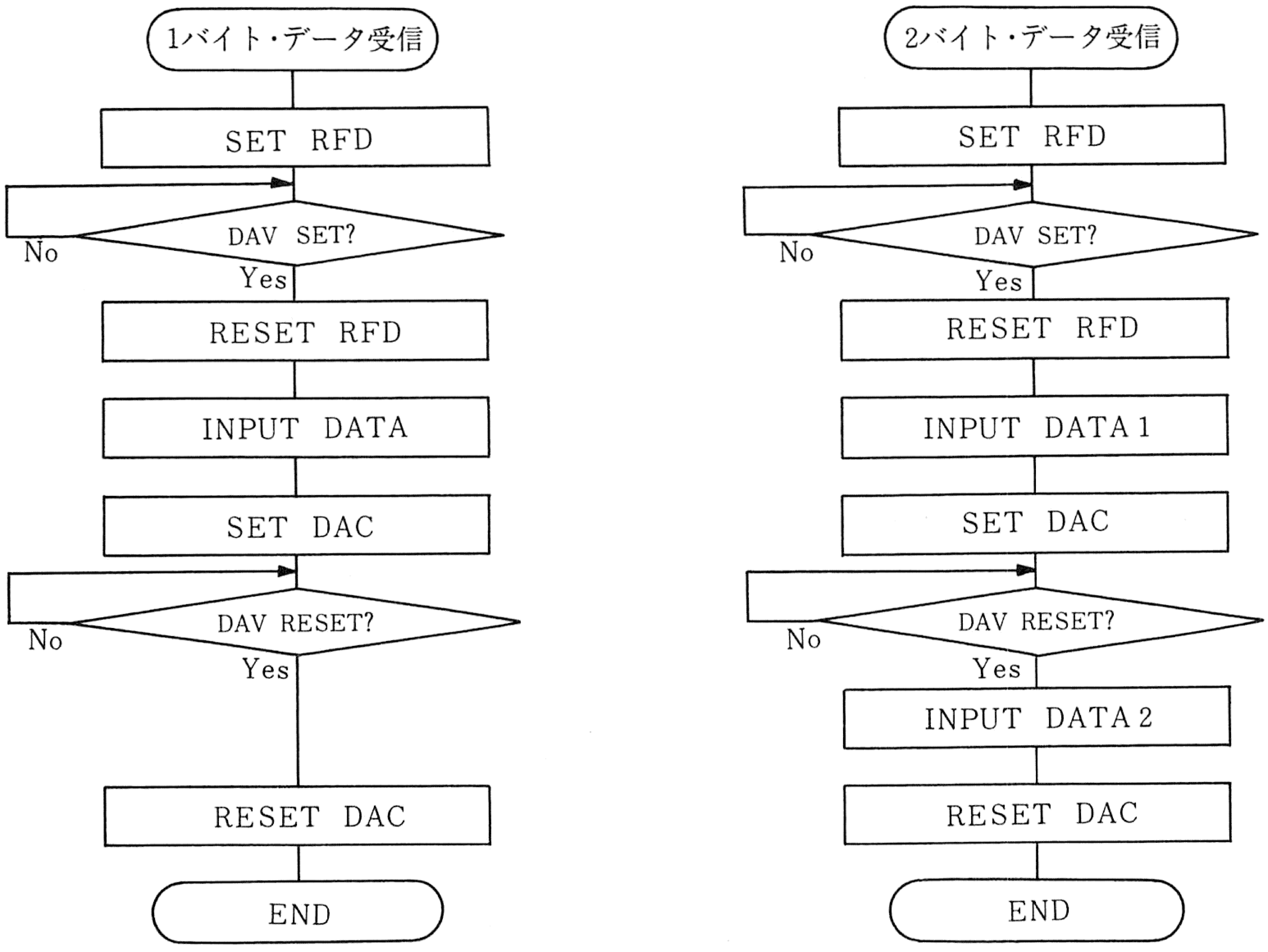
Not found!

KEY WORD 1 ?
```

4.2　ディスク・ユニット・メモリ・リード・プログラム

このプログラムを用いる事によって，ＰＣ－８０３１およびＰＣ－８０３１－２Ｗなど，インテリジェント・タイプのミニ・フロッピディスク・ユニット内に内臓されたメモリの内容を調べる事ができます．

Ｄｉｓｋ－ＢＡＳＩＣに頼らず，ディスク・ユニットとのハンドシェイクを行なう場合の参考になると思い掲載しますが，このプログラムでは，コマンド・ナンバ１１のトランスミット・コマンドによって，ディスク・ユニット内のメモリの内容を本体側に転送させています．



```
100 WIDTH 80,25:CONSOLE 0,25,0,0:PRINT CHR$(12);
110 PRINT"*****************************"
120 PRINT"                             "
130 PRINT"   8031 DISK UNIT MEMORY READ  "
140 PRINT"                             "
150 PRINT"*****************************"
160 PRINT
170 PRINT"   0000H-7FFFH : ROM AREA    7F00H-7FFFH : SYSTEM AREA"
180 PRINT"   8000H-FFFFH : RAM AREA    4000H-4FFFH : WRITING BUFFER"
190 PRINT"                             5000H-5FFFH : READING BUFFER"
200 PRINT"                             6000H-7EFFH : NONE USE"
210 PA=&HFC:PB=PA+1:PC=PA+2:CW=PA+3
220 PRINT
230 INPUT"BEGIN Address";A$:S=VAL("&H"+RIGHT$(A$,4)):IF S<0 THEN S=S+65536!
240 INPUT"E N D Address";A$:E=VAL("&H"+RIGHT$(A$,4)):IF E<0 THEN E=E+65536!
250 PRINT
260 IF S>E THEN 220
270 IF S-INT(S/16)*16<> 0 THEN S=S-1:GOTO 270
280 IF E-INT(E/16)*16<>15 THEN E=E+1:GOTO 280
290 A1=INT(S/256):A2=S-A1*256
300 A3=INT((E-S+1)/256):A4=E-S+1-A3*256
310 PRINT "BEGIN:";RIGHT$(" 0"+HEX$(A1)+RIGHT$("0"+HEX$(A2),2),5);"H"
320 PRINT "BLOCK:";RIGHT$(" 0"+HEX$(A3)+RIGHT$("0"+HEX$(A4),2),5);"H"
330 A=11:GOSUB 500 'output transmit command
340 A=A1:GOSUB 520 'output beggning address high
350 A=A2:GOSUB 520 'output beggning address low
360 A=A3:GOSUB 520 'output loading block number high
370 A=A4:GOSUB 520 'output loading block number low
380 FOR I=S TO E STEP 16
390   PRINT " ";RIGHT$("000"+HEX$(I),4);": ";:AS$=" "
400   FOR J=I TO I+15
410     GOSUB 610 'input data
420     PRINT RIGHT$("0"+HEX$(A),2);" ";
430     C$=CHR$(A)
440     IF A<&H20 OR A=&H60 OR A=&HA0 OR A>&HF7 THEN C$="."
450     AS$=AS$+C$
460   NEXT
470   PRINT AS$
480 NEXT
490 GOTO 220
500 'output command
510 OUT CW,15                          'ATN SET
520 'output data
530 IF (INP(PC) AND 2)=0 THEN 530      'RFD SET?
540 OUT CW,14                          'ATN RESET
550 OUT PB,A                           'COMMAND or DATA OUTPUT
560 OUT CW,9                           'DAV SET
570 IF (INP(PC) AND 4)=0 THEN 570      'DAC SET?
580 OUT CW,8                           'DAV RESET
590 IF (INP(PC) AND 4) THEN 590        'DAC RESET
600 RETURN
610 'input data
620 OUT CW,11                          'RFD SET
630 IF (INP(PC) AND 1)=0 THEN 630      'DAV SET?
640 OUT CW,10                          'RFD RESET
650 A=INP(PA)                          'DATA INPUT
660 OUT CW,13                          'DAC SET
670 IF (INP(PC) AND 1) THEN 670        'DAV RESET?
680 OUT CW,12                          'DAC RESET
690 RETURN
```

```
******************************

  8031 DISK UNIT MEMORY READ

******************************

  0000H-7FFFH : ROM AREA    7F00H-7FFFH : SYSTEM AREA
  8000H-FFFFH : RAM AREA    4000H-4FFFH : WRITING BUFFER
                            5000H-5FFFH : READING BUFFER
                            6000H-7EFFH : NONE USE

BEGIN Address? 5000
E N D Address? 503F

BEGIN:05000H
BLOCK: 0040H
 5000: 20 20 20 20 20 20 20 20 20 20 3A 8F E4 44 41 54            :+◢DAT
 5010: 41 20 49 4E 50 55 54 00 6A 88 94 02 9C 20 43 57  A INPUT.jl ̄.┌ CW
 5020: 2C 0F 0D 20 20 20 20 20 20 20 20 20 20 20 20 20  ,..
 5030: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 3A 8F                :+

BEGIN Address?
```

ＮＥＣのＰＣ－８０３１－２Ｗでは，従来のＰＣ－８０３１の全ての機能に加えて，両面倍密度での仕様もサポートされています．これに伴なって制御コマンドも２倍の３０種に拡張されました．

新しいコマンドの中には，ディスク・ユニットが従来の片面仕様でも使えるように，両面／片面の切り換えコマンド（コマンド・ナンバ２３）なども用意されており，ＰＣ－８０３１用として開発された多くの有益なソフト・ウェアが全てそのままで流用できるように考慮されています．

次にあげるプログラムは，ミニ・フロッピ・ディスク・ユニット内のメモリを調べるためのプログラムを，２Ｗになって拡張された，ナンバ２１のコマンドによって高速化したものです．

従来のトランスミット・コマンドでは，１度のアクセプタ・ハンドシェイクによって１バイトずつのデータしか受信する事ができませんでしたが，２Ｗでは２バイト単位でのデータ送受信が行なえるコマンドも用意されており，実際にＤｉｓｋ－ＢＡＳＩＣでも利用しています．

参考のため，従来のトランスミット・コマンドによるデータの受信と，２Ｗのもののフロー・チャートを対比して記載しておきます．

```
100 WIDTH 80,25:CONSOLE 0,25,0,0:PRINT CHR$(12);
110 PRINT"*******************************"
120 PRINT"                               "
130 PRINT"  2Way DISK UNIT MEMORY READ   "
140 PRINT"                               "
150 PRINT"*******************************"
160 PRINT
170 PRINT"  0000H-7FFFH : ROM AREA    7F00H-7FFFH : SYSTEM AREA"
180 PRINT"  8000H-FFFFH : RAM AREA    4000H-4FFFH : WRITING BUFFER"
190 PRINT"                            5000H-5FFFH : READING BUFFER"
200 PRINT"                            6000H-7EFFH : NONE USE"
210 PA=&HFC:PB=PA+1:PC=PA+2:CW=PA+3
220 PRINT
230 INPUT"BEGIN Address";A$:S=VAL("&H"+RIGHT$(A$,4)):IF S<0 THEN S=S+65536!
240 INPUT"E N D Address";A$:E=VAL("&H"+RIGHT$(A$,4)):IF E<0 THEN E=E+65536!
250 PRINT
260 IF S>E THEN 220
270 IF S-INT(S/16)*16<> 0 THEN S=S-1:GOTO 270
280 IF E-INT(E/16)*16<>15 THEN E=E+1:GOTO 280
290 A1=INT(S/256):A2=S-A1*256
300 A3=INT((E-S+1)/256):A4=E-S+1-A3*256
310 PRINT "BEGIN:";RIGHT$(" 0"+HEX$(A1)+RIGHT$("0"+HEX$(A2),2),5);"H"
320 PRINT "BLOCK:";RIGHT$(" 0"+HEX$(A3)+RIGHT$("0"+HEX$(A4),2),5);"H"
330 A=21:GOSUB 540 'output transmit command
340 A=A1:GOSUB 560 'output beggning address high
350 A=A2:GOSUB 560 'output beggning address low
360 A=A3:GOSUB 560 'output loading block number high
370 A=A4:GOSUB 560 'output loading block number low
380 FOR I=S TO E STEP 16
390   PRINT " ";RIGHT$("000"+HEX$(I),4);": ";:AS$=" "
400   FOR J=I TO I+15 STEP 2
410     GOSUB 650 'input datas
420     PRINT RIGHT$("0"+HEX$(A1),2);" ";
430     C$=CHR$(A1)
440     IF A1<&H20 OR A1=&H60 OR A1=&HA0 OR A1>&HF7 THEN C$="."
450     AS$=AS$+C$
460     PRINT RIGHT$("0"+HEX$(A2),2);" ";
470     C$=CHR$(A2)
480     IF A2<&H20 OR A2=&H60 OR A2=&HA0 OR A2>&HF7 THEN C$="."
490     AS$=AS$+C$
500   NEXT
510   PRINT AS$
520 NEXT
530 GOTO 220
540 'output command
550 OUT CW,15                             'ATN SET
560 'output data
570 IF (INP(PC) AND 2)=0 THEN 570         'RFD SET?
580 OUT CW,14                             'ATN RESET
590 OUT PB,A                              'COMMAND or DATA OUTPUT
600 OUT CW,9                              'DAV SET
610 IF (INP(PC) AND 4)=0 THEN 610         'DAC SET?
620 OUT CW,8                              'DAV RESET
630 IF (INP(PC) AND 4) THEN 630           'DAC RESET
640 RETURN
650 'input data
660 OUT CW,11                             'RFD SET
670 IF (INP(PC) AND 1)=0 THEN 670         'DAV SET?
680 OUT CW,10                             'RFD RESET
690 A1=INP(PA)                            'DATA1 INPUT
700 OUT CW,13                             'DAC SET
710 IF (INP(PC) AND 1) THEN 710           'DAV RESET?
720 A2=INP(PA)                            'DATA2 INPUT
730 OUT CW,12                             'DAC RESET
740 RETURN
```

```
******************************

  2Way DISK UNIT MEMORY READ

******************************

  0000H-7FFFH : ROM AREA   7F00H-7FFFH : SYSTEM AREA
  8000H-FFFFH : RAM AREA   4000H-4FFFH : WRITING BUFFER
                           5000H-5FFFH : READING BUFFER
                           6000H-7EFFH : NONE USE

BEGIN Address? 0
E N D Address? 3F

BEGIN: 0000H
BLOCK: 0040H
 0000: C3 7E 00 00 00 00 00 00 F3 CD 22 7F C3 4E 00 00  ﾃ~......ﾙﾍﾞ ﾃN..
 0010: C3 D9 06 00 00 00 00 00 C3 0F 07 00 00 00 00 00  ﾃﾙ......ﾃ.......
 0020: C3 A7 02 00 C3 D9 06 C3 0F 07 C3 C1 00 C3 D5 02  ﾃｧ..ﾃﾙ.ﾃ..ﾃﾁ.ﾃﾕ.
 0030: C3 D6 02 C3 96 03 C3 96 01 C3 E7 01 C3 C2 02 C3  ﾃﾖ.ﾃ|.ﾃ|.ﾃ◤.ﾃﾂ.ﾃ

BEGIN Address?
```

4.3 MEMORY TO DATA

一般に高速性を要するプログラムを開発して行く場合，N－BASICだけでは，どうしても思ったとうりの実行速度を得られない事があります．

そこで，本書の主旨でもあるマシン・ランゲージの使用を考えるのですが，当然全てをマシン語でコーディングする必要は無く，最も高速化の必要な部分のみをサブルーチンとして，N－BASICから呼びだすのが普通だと思います．これは，入出力の部分などは容易に修正の行なえる方が良く，N－BASICとくにDisk－BASICではマシン語が不利な立場に置かれているために，保守性の面からも言える事です．

以上の点から，メモリ上のマシン語プログラムをN－BASICのデータとして組み込む事が必要になるのですが，この作業を自動的に行なうのが，ここで紹介するプログラムです．

4.3.1 使用方法

まず先頭アドレスから実行しますが，プログラム自体は完全リロケータブルですので，メモリ上のどのアドレスに転送して使う事もできます．ただし，プログラム末の6バイトをワーキング・エリアとして利用しますので変更が必要な場合があるかも知れません．

プログラムを実行すると，プロンプトマークが表示されますので，メモリ上のデータのスタート・アドレスとエンド・アドレスをコンマで区切って入力してください．入力時に

エラーが有ればマシン語モニタに制御が移ります．

Ｎ－ＢＡＳＩＣのデータ文は，６５０００行から１行おきにつくられて行き，実行以前に何らかのプログラムが入っていた場合にはその後にアペンドされます．必要なら実行後にリナンバをかけて行番号をそろえてください．

なお，３２Ｋのシステムでも８Ｋバイト以上のマシン語をデータにする事はメモリの限界なので不可能です．８Ｋバイト近いマシン語でも１～２秒でデータが生成されるはずですから，数秒間反応がない場合にはメモリの容量を越えて暴走しているかも知れません．

また，このプログラムではＮ－ＢＡＳＩＣのテキスト自身を書き換え，ＲＯＭ内のサブルーチンやワーキング・エリアを活用していますので，解析，改良を行なえば多くの応用ができると思います．

```
                    ;******************************
                    ;
                    ;    MEMORY TO DATA VER 2.0
                    ;
                    ;******************************
                    ;
0000                EOLMRK:EQU  0               ; END OF LINE MARK
0020                SPCCOD:EQU  ' '             ; CODE OF SPACE
002C                COMCOD:EQU  ','             ; CODE OF COMMA
005D                PROMPT:EQU  ']'             ; PROMPT MARK
0084                DATCOD:EQU  84H             ; CODE OF DATA
FDE8                INITNO:EQU  65000           ; INITIAL LINE NUMBER
                    ;
0257                DSPCHR:EQU  0257H           ; DISPLAY CHARACTER
1F8B                WRKSET:EQU  1F8BH           ; SET WORKING AREAS
3D76                SEARCH:EQU  3D76H           ; SEARCH EOF OF TEXT
5C5E                MONERR:EQU  5C5EH           ; MONITOR ERROR
5E21                HEX4IN:EQU  5E21H           ; INPUT ADDRESS
5E83                HEXASC:EQU  5E83H           ; HEX TO ASCII CODE
5ED3                HLCPDE:EQU  5ED3H           ; COMPARE HL,DE
5FCA                CRORLF:EQU  5FCAH           ; DISPLAY CR AND LF
                    ;
                           ORG  0EE00H
                    ;
EE00 21E8FD                LD   HL,INITNO
EE03 228DEE                LD   (LINENO),HL
                    ;
EE06 3E5D                  LD   A,PROMPT
EE08 CD5702                CALL DSPCHR
                    ;
EE0B 210000                LD   HL,0
EE0E 54                    LD   D,H
EE0F 5D                    LD   E,L
EE10 CD215E                CALL HEX4IN
EE13 CA5E5C                JP   Z,MONERR
EE16 EB                    EX   DE,HL
EE17 CD215E                CALL HEX4IN
EE1A C25E5C                JP   NZ,MONERR
EE1D CDD35E                CALL HLCPDE
EE20 DA5E5C                JP   C,MONERR
EE23 228BEE                LD   (ENDADR),HL
                    ;
EE26 D5                    PUSH DE
EE27 CD763D                CALL SEARCH
EE2A EB                    EX   DE,HL
```

```
EE2B D1                 POP   DE
                ;
EE2C 2289EE     TOLINE:LD    (BASSTK),HL
EE2F 23                INC   HL
EE30 23                INC   HL
EE31 ED4B8DEE          LD    BC,(LINENO)
EE35 71                LD    (HL),C
EE36 23                INC   HL
EE37 70                LD    (HL),B
EE38 23                INC   HL
EE39 03                INC   BC
EE3A ED438DEE          LD    (LINENO),BC
EE3E 3684              LD    (HL),DATCOD
EE40 23                INC   HL
EE41 3620              LD    (HL),SPCCOD
EE43 23                INC   HL
                ;
EE44 0610              LD    B,16
EE46 D5         NEWMEM:PUSH  DE
EE47 1A                LD    A,(DE)
EE48 CD835E            CALL  HEXASC
EE4B 72                LD    (HL),D
EE4C 23                INC   HL
EE4D 73                LD    (HL),E
EE4E 23                INC   HL
EE4F D1                POP   DE
EE50 E5                PUSH  HL
EE51 2A8BEE            LD    HL,(ENDADR)
EE54 CDD35E            CALL  HLCPDE
EE57 E1                POP   HL
EE58 2816              JR    Z,EOFILE
EE5A 13                INC   DE
EE5B 362C              LD    (HL),COMCOD
EE5D 23                INC   HL
EE5E 10E6              DJNZ  NEWMEM
                ;
EE60 2B                DEC   HL
EE61 3600              LD    (HL),EOLMRK
EE63 23                INC   HL
EE64 DD2A89EE          LD    IX,(BASSTK)
EE68 DD7500            LD    (IX),L
EE6B DD7401            LD    (IX+1),H
EE6E 18BC              JR    TOLINE
                ;
EE70 3600       EOFILE:LD    (HL),EOLMRK
EE72 23                INC   HL
EE73 DD2A89EE          LD    IX,(BASSTK)
EE77 DD7500            LD    (IX),L
EE7A DD7401            LD    (IX+1),H
EE7D 3600              LD    (HL),EOLMRK
EE7F 23                INC   HL
EE80 3600              LD    (HL),EOLMRK
EE82 23                INC   HL
EE83 CDCA5F            CALL  CRORLF
EE86 C38B1F            JP    WRKSET
                ;
EE89            BASSTK:DS    2              ; BASIC TOP OF LINE STACK
EE8B            ENDADR:DS    2              ; MEMORY END ADDRESS
EE8D            LINENO:DS    2              ; BASIC DATAS LINE NUMBER
```

```
*GEE00
]0,FF
Ok


list
65000 DATA F3,31,FF,FF,C3,3B,00,00,C3,6A,00,C3,57,17,AB,F0
65001 DATA C3,59,42,C3,6A,00,DA,0C,C3,A6,40,F3,0B,C3,7E,50
65002 DATA C3,DA,F1,C3,9C,27,88,0C,C3,DD,F1,C3,60,0D,46,0C
65003 DATA C3,E0,F1,9F,0F,C3,57,02,C3,E3,F1,AF,32,75,EA,CD
65004 DATA F1,0C,30,C7,18,CD,00,00,00,00,00,00,00,00,00,00
65005 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
65006 DATA 00,00,00,00,00,00,00,00,00,00,F3,CD,4C,16,AF,32
65007 DATA 60,EA,3D,32,61,EA,01,81,00,C5,01,00,FF,C5,C3,F6
65008 DATA 08,CD,B0,F1,3A,55,EB,ED,47,3E,FF,D3,E4,32,55,EA
65009 DATA ED,5E,FB,01,81,3C,C3,10,3C,C5,D5,F5,CD,B3,F1,32
65010 DATA B9,ED,3E,01,38,02,3E,02,CD,21,01,F1,F5,47,38,0F
65011 DATA 28,10,CD,5A,01,E6,40,CA,68,01,CD,39,01,18,03,CD
65012 DATA 4C,01,CD,5A,01,F1,D1,C1,B7,C9,C9,DB,40,E6,08,EE
65013 DATA 08,C9,C5,D5,F5,3A,67,EA,E6,FE,D3,40,F6,01,D3,40
65014 DATA 32,67,EA,CD,CB,00,28,11,3E,91,CD,29,02,AF,32,C5
65015 DATA ED,3C,32,CB,ED,CD,FF,00,A7,C1,F5,20,C5,18,C6,3E
Ok
```

4.4　テープ・オート・スタート・プログラム

オート・スタートのテープをつくるためのプログラムで，実際にオート・スタートを行なう部分と，プログラムをオート・スタート形式でセーブするための部分から構成されます．

このプログラムでは，プログラマブル・ファンクション・キーの特色を活用し，ファンクション・キー・バッファ，ファンクション・キー・フラグ，ファンクション・キー・ポインタなどをコントロールする事によって，オート・スタートを実現しています．

4.4.1　使用方法

オート・スタート・プログラムはＥＣＣ０Ｈ番地からのインプット・バッファ内に位置していますが，この部分はオート・スタートしたいプログラムの先頭に付属してセーブされ，ロード時には一番最初に実行されます．このプログラムが実行される事によって，オート・スタートを行なうべきプログラム本体もロード，実行されるのですから，セーブするプログラムのファイル・ネームやアドレスなどによって，データを修正しなくてはなりません．

また，セーバーの方のデータも，ターゲット・プログラムのファイル・ネームやアドレスによって変更しておきます．

プログラムのセーブは，セーバーを実行する事によって自動的に行なう事ができ，ロードは，リセットの後，マシン語モニタのＬコマンドによって行ないます．

データのつくり方によっては，ベーシック，マシン語などを混合した複数のプログラムを一度でロード，実行する事ができます．

また，オート・スタート・プログラムをＩＰＬに組み込んでしまえば，Ｄｉｓｋ—ＢＡＳＩＣのオート・スタートにも簡単に応用する事ができます．いろいろと試してみてください．

```
                    ;*****************************
                    ;
                    ;        TAPE AUTO START
                    ;
                    ;*****************************
                    ;
                    ;
0081                JMPBAS:EQU  0081H
5C66                JMPMON:EQU  5C66H
                    ;
EA68                FKFLAG:EQU  0EA68H
EABC                FK5BGN:EQU  0EABCH
EDC0                FPOINT:EQU  0EDC0H
                    ;
                    ;
                           ORG  0ECC0H
                    ;
                    ;
ECC0 217275                LD   HL,7572H
ECC3 22BCEA                LD   (FK5BGN),HL
ECC6 216E0D                LD   HL,0D6EH
ECC9 22BEEA                LD   (FK5BGN+2),HL
ECCC AF                    XOR  A
ECCD 32C0EA                LD   (FK5BGN+4),A
ECD0 32CCEA                LD   (FK5BGN+16),A
                    ;
ECD3 21E1EC                LD   HL,KEY
ECD6 22C0ED                LD   (FPOINT),HL
ECD9 3E01                  LD   A,1
ECDB 3268EA                LD   (FKFLAG),A
ECDE C38100                JP   JMPBAS
                    ;
                    ;
ECE1 636C6F61  KEY:        DB   'cload"FILNAM"',13
ECE5 64224649
ECE9 4C4E414D
ECED 220D
ECEF 72756E0D              DB   'run',13,0
ECF3 00
```

```
                    ;
                    ;
                    ;
                    ;*****************************
                    ;
                    ;       AUTO START SAVER
                    ;
                    ;*****************************
                    ;
                    ;
                          ORG   0E9A0H
                    ;
                    ;
E9A0 21B9E9               LD    HL,FKLOAD
E9A3 11BCEA               LD    DE,FK5BGN
E9A6 010F00               LD    BC,15
E9A9 EDB0                 LDIR
                    ;
E9AB 21C9E9               LD    HL,FKSAVE
E9AE 22C0ED               LD    (FPOINT),HL
E9B1 3E01                 LD    A,1
E9B3 3268EA               LD    (FKFLAG),A
E9B6 C3665C               JP    JMPMON
                    ;
                    ;
E9B9 0D0D0D0D FKLOAD:DB   13,13,13,13,13,13,13
E9BD 0D0D0D
E9C0 6C0D6765             DB    'l',13,'gecc0',13,0
E9C4 6363300D
E9C8 00
                    ;
E9C9 77656136 FKSAVE:DB   'wea68,eaca',13
E9CD 382C6561
E9D1 63610D
E9D4 77656363             DB    'wecc0,ed95',13
E9D8 302C6564
E9DC 39350D
E9DF 02637361             DB    2,'csave"FILNAM"',13,0
E9E3 76652246
E9E7 494C4E41
E9EB 4D220D00
```

5
付　録

5.1 キー・ワード，中間言語，エントリ・アドレス対応表

5.1.1 N－BASIC

AND		F8		
ABS	FF	86		2671
ATN	FF	8E		3372
ASC	FF	95		5498
AUTO		A9		46CF
ATTR$		E7		
BCD$	FF	9E	F120 ->	1875
BEEP		B2		0D41
CONSOLE		9F		0884
CLOSE		CA	F135 ->	1875
CONT		99		4383
CLEAR		92		44E8
CLOAD		9B		1F10
CSAVE		9A		1EC0
CSRLIN		E6		
CINT	FF	9F		277F
CSNG	FF	A0		27B3
CDBL	FF	A1		27DF
CVI	FF	A3	F0E1 ->	1875
CVS	FF	A4	F0E4 ->	1875
CVD	FF	A5	F0E7 ->	1875
COS	FF	8C		32F6
CHR$	FF	96		54A8
CMD		B8	F0FC ->	1875
COLOR		B5		0951
DATA		84		45BE
DIM		86		4E37
DEFSTR		AB		445B
DEFINT		AC		445E
DEFSNG		AD		4461
DEFDBL		AE		4464
DSKO$		C2	F12C ->	1875
DEF		97		50CC
DELETE		A8		58D9
DSKI$		E8		
DSKF	FF	A6	F159 ->	1875
DEC	FF	9D	F123 ->	1875
DATE$		EB		
END		81		432F
ELSE		A1		45C0
ERASE		A5		43DF
ERROR		A6		46C4
ERL		DF		
ERR		E0		
EXP	FF	8B		31F3
EOF	FF	A7	F0EA ->	1875

EQV		FB		
FORMAT		B3	F12F ->	1875
FOR		82		4159
FIELD		C6	F102 ->	1875
FILES		CD	F14D ->	1875
FN		DC		
FRE	FF	8F		5051
FIX	FF	A2		282C
FPOS	FF	AA	F168 ->	1875
GOTO		89		456D
GO TO		89		456D
GOSUB		8D		4555
GET		C7		1886
HEX$	FF	9A		5275
INPUT		85		48DA
IF		8B		4702
INSTR		E3		
INT	FF	85		283F
INP	FF	90		56A1
IMP		FC		
INIT		D4		2262
INKEY$		E9		
ISET		BD	F105 ->	1875
IRESET		BE	F108 ->	1875
IEEE	FF	EC		
KILL		CF	F141 ->	1875
KEY		B4		1343
LET		88		45DE
LOCATE		D5		0792
LINE		AF		4877
LOAD		CB	F138 ->	1875
LSET		D0	F144 ->	1875
LPRINT		9D		473A
LLIST		9E		5707
LPOS	FF	9B		5074
LISTEN		C1	F10E ->	1875
LIST		93		570C
LFILES		D3	F126 ->	1875
LOG	FF	8A		2503
LOC	FF	A8	F0ED ->	1875
LEN	FF	92		548C
LEFT$	FF	81		54F9
LOF	FF	A9	F0F0 ->	1875
MOUNT		C4	F153 ->	1875
MERGE		CC	F13B ->	1875
MOD		FD		

MKI$	FF	AB	F0F3 ->	1875
MKS$	FF	AC	F0F6 ->	1875
MKD$	FF	AD	F0F9 ->	1875
MID$	FF	83		5532
MOTOR		B9		0DA1
MON		B7		0D5D
MAT		C0	F111 ->	1875
NEXT		83		4A08
NAME		CE	F13E ->	1875
NEW		94		3DE0
NOT		DE		
OUT		9C		56AD
ON		95		4642
OPEN		C5	F0FF ->	1875
OR		F9		
OCT$	FF	99		5270
PUT		C8		1891
POKE		98		5918
PRINT		91		4742
POS	FF	91		5079
PEEK	FF	97		5911
PORT	FF	9C		20F9
POLL		BA	F114 ->	1875
PSET		B1		06B8
PRESET		B0		0705
POINT		EF		
READ		87		4939
RUN		8A		453D
RESTORE		8C		4302
RETURN		8E		45A3
REMOVE		C3	F150 ->	1875
REM		8F		45C0
RESUME		A7		468C
RSET		D1	F147 ->	1875
RIGHT$	FF	82		5529
RND	FF	88		3283
RENUM		AA		5AED
RBYTE		BB	F11A ->	1875
STOP		90		432A
SWAP		A4		439C
SET		C9	F156 ->	1875
SAVE		D2	F14A ->	1875
SPC(		DD		
STEP		DA		
SGN	FF	84		2686
SQR	FF	87		31A1
SIN	FF	89		32FC
STR$	FF	93		527A
STRING$		E1		
SPACE$	FF	98		54DF
STATUS		EE		
SRQ		ED		
TRON		A2		4396
TROFF		A3		4397
TAB(		D9		
TO		D7		
THEN		D8		
TAN	FF	8D		335D
TERM		B6		0DB8
TALK		BF	F10B ->	1875
TIME$		EA		
USING		E2		
USR		DB		
VAL	FF	94		5553
VARPTR		E5		
WIDTH		A0		0843
WAIT		96		56B3
WBYTE		BC	F117 ->	1875
XOR		FA		
+		F3		
-		F4		
*		F5		
/		F6		
^		F7		
¥		FE		
'		E4		
>		F0		
=		F1		
<		F2		

5.1.2 Disk-version [14-Nov-1980]

```
Disk version [14-Nov-1980]
How many files(0-15)?
```

AND		F8		
ABS	FF	86		2671
ATN	FF	8E		3372
ASC	FF	95		5498
AUTO		A9		46CF
ATTR$		E7		
BCD$	FF	9E	F120 ->	1875
BEEP		B2		0D41
CONSOLE		9F		0884
CLOSE		CA	F135 ->	E240
CONT		99		4383
CLEAR		92		44E8
CLOAD		9B		1F10
CSAVE		9A		1EC0
CSRLIN		E6		
CINT	FF	9F		277F
CSNG	FF	A0		27B3
CDBL	FF	A1		27DF
CVI	FF	A3	F0E1 ->	E1EB
CVS	FF	A4	F0E4 ->	E1EE
CVD	FF	A5	F0E7 ->	E1F1
COS	FF	8C		32F6
CHR$	FF	96		54A8
CMD		B8	F0FC ->	1875
COLOR		B5		0951
DATA		84		45BE
DIM		86		4E37
DEFSTR		AB		445B
DEFINT		AC		445E
DEFSNG		AD		4461
DEFDBL		AE		4464
DSKO$		C2	F12C ->	E47C
DEF		97		50CC
DELETE		A8		58D9
DSKI$		E8		
DSKF	FF	A6	F159 ->	E812
DEC	FF	9D	F123 ->	1875
DATE$		EB		
END		81		432F
ELSE		A1		45C0
ERASE		A5		43DF
ERROR		A6		46C4
ERL		DF		
ERR		E0		
EXP	FF	8B		31F3
EOF	FF	A7	F0EA ->	E84C
EQV		FB		
FORMAT		B3	F12F ->	D804
FOR		82		4159
FIELD		C6	F102 ->	E0BF
FILES		CD	F14D ->	E284
FN		DC		
FRE	FF	8F		5051
FIX	FF	A2		282C
FPOS	FF	AA	F168 ->	E868
GOTO		89		456D
GO TO		89		456D
GOSUB		8D		4555

GET		C7		1886
HEX$	FF	9A		5275
INPUT		85		48DA
IF		8B		4702
INSTR		E3		
INT	FF	85		283F
INP	FF	90		56A1
IMP		FC		
INIT		D4		2262
INKEY$		E9		
ISET		BD	F105 ->	1875
IRESET		BE	F108 ->	1875
IEEE	FF	EC		
KILL		CF	F141 ->	DDDE
KEY		B4		1343
LET		88		45DE
LOCATE		D5		0792
LINE		AF		4877
LOAD		CB	F138 ->	DDF0
LSET		D0	F144 ->	E012
LPRINT		9D		473A
LLIST		9E		5707
LPOS	FF	9B		5074
LISTEN		C1	F10E ->	1875
LIST		93		570C
LFILES		D3	F126 ->	E27F
LOG	FF	8A		2503
LOC	FF	A8	F0ED ->	E823
LEN	FF	92		548C
LEFT$	FF	81		54F9
LOF	FF	A9	F0F0 ->	E83B
MOUNT		C4	F153 ->	E208
MERGE		CC	F13B ->	DDF1
MOD		FD		
MKI$	FF	AB	F0F3 ->	E1D2
MKS$	FF	AC	F0F6 ->	E1D5
MKD$	FF	AD	F0F9 ->	E1D8
MID$	FF	83		5532
MOTOR		B9		0DA1
MON		B7		0D5D
MAT		C0	F111 ->	1875
NEXT		83		4A08
NAME		CE	F13E ->	DB84
NEW		94		3DE0
NOT		DE		
OUT		9C		56AD
ON		95		4642
OPEN		C5	F0FF ->	DBD6
OR		F9		
OCT$	FF	99		5270
PUT		C8		1891
POKE		98		5918
PRINT		91		4742
POS	FF	91		5079
PEEK	FF	97		5911
PORT	FF	9C		20F9
POLL		BA	F114 ->	1875
PSET		B1		06B8
PRESET		B0		0705
POINT		EF		
READ		87		4939
RUN		8A		453D
RESTORE		8C		4302
RETURN		8E		45A3
REMOVE		C3	F150 ->	E20D
REM		8F		45C0
RESUME		A7		468C
RSET		D1	F147 ->	E011
RIGHT$	FF	82		5529
RND	FF	88		3283
RENUM		AA		5AED
RBYTE		BB	F11A ->	1875
STOP		90		432A
SWAP		A4		439C
SET		C9	F156 ->	E109
SAVE		D2	F14A ->	DE3D
SPC(		DD		
STEP		DA		
SGN	FF	84		2686
SQR	FF	87		31A1
SIN	FF	89		32FC
STR$	FF	93		527A
STRING$		E1		
SPACE$	FF	98		54DF
STATUS		EE		
SRQ		ED		
TRON		A2		4396
TROFF		A3		4397
TAB(		D9		
TO		D7		
THEN		D8		
TAN	FF	8D		335D
TERM		B6		0DB8
TALK		BF	F10B ->	1875
TIME$		EA		
USING		E2		
USR		DB		
VAL	FF	94		5553
VARPTR		E5		
WIDTH		A0		0843
WAIT		96		56B3
WBYTE		BC	F117 ->	1875
XOR		FA		
+		F3		
-		F4		
*		F5		
/		F6		
^		F7		
¥		FE		
'		E4		
>		F0		
=		F1		
<		F2		

5.1.3 Two surface disk version [20-Sep-1981]

```
Two surface disk version [20-Sep-1981]
How many files(0-15)?
```

AND		F8		
ABS	FF	86		2671
ATN	FF	8E		3372
ASC	FF	95		5498
AUTO		A9		46CF
ATTR$		E7		
BCD$	FF	9E	F120 ->	1875
BEEP		B2		0D41
CONSOLE		9F		0884
CLOSE		CA	F135 ->	E200
CONT		99		4383
CLEAR		92		44E8
CLOAD		9B		1F10
CSAVE		9A		1EC0
CSRLIN		E6		
CINT	FF	9F		277F
CSNG	FF	A0		27B3
CDBL	FF	A1		27DF
CVI	FF	A3	F0E1 ->	E1AB
CVS	FF	A4	F0E4 ->	E1AE
CVD	FF	A5	F0E7 ->	E1B1
COS	FF	8C		32F6
CHR$	FF	96		54A8
CMD		B8	F0FC ->	1875
COLOR		B5		0951
DATA		84		45BE
DIM		86		4E37
DEFSTR		AB		445B
DEFINT		AC		445E
DEFSNG		AD		4461
DEFDBL		AE		4464
DSKO$		C2	F12C ->	E43C
DEF		97		50CC
DELETE		A8		58D9
DSKI$		E8		
DSKF	FF	A6	F159 ->	E80C
DEC	FF	9D	F123 ->	1875
DATE$		EB		
END		81		432F
ELSE		A1		45C0
ERASE		A5		43DF
ERROR		A6		46C4
ERL		DF		
ERR		E0		
EXP	FF	8B		31F3
EOF	FF	A7	F0EA ->	E846
EQV		FB		
FORMAT		B3	F12F ->	D774
FOR		82		4159
FIELD		C6	F102 ->	E07F
FILES		CD	F14D ->	E244
FN		DC		
FRE	FF	8F		5051
FIX	FF	A2		282C

FPOS	FF	AA	F168 ->	E862
GOTO		89		456D
GO TO		89		456D
GOSUB		8D		4555
GET		C7		1886
HEX$	FF	9A		5275
INPUT		85		48DA
IF		8B		4702
INSTR		E3		
INT	FF	85		283F
INP	FF	90		56A1
IMP		FC		
INIT		D4		2262
INKEY$		E9		
ISET		BD	F105 ->	1875
IRESET		BE	F108 ->	1875
IEEE	FF	EC		
KILL		CF	F141 ->	DD9E
KEY		B4		1343
LET		88		45DE
LOCATE		D5		0792
LINE		AF		4877
LOAD		CB	F138 ->	DDB0
LSET		D0	F144 ->	DFD2
LPRINT		9D		473A
LLIST		9E		5707
LPOS	FF	9B		5074
LISTEN		C1	F10E ->	1875
LIST		93		570C
LFILES		D3	F126 ->	E23F
LOG	FF	8A		2503
LOC	FF	A8	F0ED ->	E81D
LEN	FF	92		548C
LEFT$	FF	81		54F9
LOF	FF	A9	F0F0 ->	E835
MOUNT		C4	F153 ->	E1C8
MERGE		CC	F13B ->	DDB1
MOD		FD		
MKI$	FF	AB	F0F3 ->	E192
MKS$	FF	AC	F0F6 ->	E195
MKD$	FF	AD	F0F9 ->	E198
MID$	FF	83		5532
MOTOR		B9		0DA1
MON		B7		0D5D
MAT		C0	F111 ->	1875
NEXT		83		4A08
NAME		CE	F13E ->	DB31
NEW		94		3DE0
NOT		DE		
OUT		9C		56AD
ON		95		4642
OPEN		C5	F0FF ->	DB83
OR		F9		
OCT$	FF	99		5270
PUT		C8		1891
POKE		98		5918

PRINT		91		4742
POS	FF	91		5079
PEEK	FF	97		5911
PORT	FF	9C		20F9
POLL		BA	F114 ->	1875
PSET		B1		06B8
PRESET		B0		0705
POINT		EF		
READ		87		4939
RUN		8A		453D
RESTORE		8C		4302
RETURN		8E		45A3
REMOVE		C3	F150 ->	E1CD
REM		8F		45C0
RESUME		A7		468C
RSET		D1	F147 ->	DFD1
RIGHT$	FF	82		5529
RND	FF	88		3283
RENUM		AA		5AED
RBYTE		BB	F11A ->	1875
STOP		90		432A
SWAP		A4		439C
SET		C9	F156 ->	E0C9
SAVE		D2	F14A ->	DDFD
SPC(		DD		
STEP		DA		
SGN	FF	84		2686
SQR	FF	87		31A1
SIN	FF	89		32FC
STR$	FF	93		527A
STRING$		E1		
SPACE$	FF	98		54DF
STATUS		EE		
SRQ		ED		
TRON		A2		4396
TROFF		A3		4397
TAB(		D9		
TO		D7		
THEN		D8		
TAN	FF	8D		335D
TERM		B6		0DB8
TALK		BF	F10B ->	1875
TIME$		EA		
USING		E2		
USR		DB		
VAL	FF	94		5553
VARPTR		E5		
WIDTH		A0		0843
WAIT		96		56B3
WBYTE		BC	F117 ->	1875
XOR		FA		
+		F3		
-		F4		
*		F5		
/		F6		
^		F7		

¥	FE
´	E4
>	F0
=	F1
<	F2

5.2

N-BASIC
リファレンス活用表

コマンド

AUTO

機　能　行番号を自動的に発生する.

書　式　AUTO〔〈行番号〉〕〔,〈増分〉〕

文　例　AUTO　100,　5

CLOAD／CLOAD?

機　能　カセットテープからメモリへプログラムをロードする.

書　式　CLOAD"〈ファイル名〉"

　　　　CLOAD?"〈ファイル名〉"

文　例　CLOAD"DEMO"

　　　　CLOAD?"DEMO"

CONT

機　能　ストップキー入力，またはSTOP文によって停止したプログラムの実行を再開する.

書　式　CONT

文　例　CONT

CSAVE

機　能　メモリ上にあるプログラムをカセットテープへセーブする.

書　式　CSAVE"〈ファイル名〉"

文　例　CSAVE"DEMO"

DELETE

機　能　プログラムの部分削除.

書　式　DELETE〔〈始点行番号〉〕〔－〈終点行番号〉〕

文　例　DELETE 10－230

FILES／LFILES

機　能　ディスクに入っているファイルの名前，種類と大きさを表示する.

書　式　1）FILES〔〈ドライブ番号〉〕

　　　　2）LFILES〔〈ドライブ番号〉〕

文　例　　FILES 2

FORMAT（ディスク）

機　能　ディスケットを初期化（フォーマット）する．

書　式　FORMAT〈ドライブ番号〉

文　例　FORMAT

KEY LIST

機　能　ファンクションキーの内容をリストする．

書　式　KEY LIST

文　例　KEY LIST

LIST／LLIST

機　能　メモリにあるプログラムの全部または一部をリストアウトする．

書　式　1）LIST〔〈行番号〉〕〔－〈行番号〉〕

　　　　2）LLIST〔〈行番号〉〕〔－〈行番号〉〕

文　例　LIST 100－200

LOAD（ディスク）

機　能　プログラムをメモリにロードする．

書　式　LOAD〈ファイル名〉〔, R〕

文　例　LOAD "FILNAM"

MERGE（ディスク）

機　能　プログラムを混合する．

書　式　MERGE〈ファイル名〉

文　例　MERGE "2：TEST"

MON

機　能　内蔵の機械語モニタに制御を移す．

書　式　MON

文　例　MON

MOUNT（ディスク）

機　能　ディスクのファイル配置表を読んで以後のディスク入出力を可能にする.

書　式　MOUNT〔〈ドライブ番号〉〔,〈ドライブ番号〉…〕〕

文　例　MOUNT

NEW

機　能　メモリにあるプログラムを抹消し，すべての変数をクリアする.

書　式　NEW

文　例　NEW

REMOVE（ディスク）

機　能　ディスクにファイル配置表を書き込んでディスク入出力を終了する.

書　式　REMOVE〔〈ドライブ番号〉〔,〈ドライブ番号〉…〕〕

文　例　REMOVE1

RENUM

機　能　プログラムの行番号を整理する.

書　式　RENUM〔〈新行番号〉〕〔,〈旧行番号〉〕〔,〈増分〉〕

文　例　RENUM

RUN

機　能　メモリにあるプログラムの実行を開始する.

ディスクからファイルをメモリにロードし，そのプログラムを実行する.

書　式　1）RUN〔〈行番号〉〕

2）RUN〈ファイル名〉〔,R〕

文　例　1）RUN

2）RUN"2：TEST"

SAVE（ディスク）

機　能　メモリのBASICプログラムをファイルにセーブする.

書　式　SAVE"〈ファイル名〉"〔,A〕

文　例　SAVE "2：MYPROG", A

TERM

機　能　システムをターミナルモードにする．

書　式　TERM〈コード〉，〈パリティ〉，〈ボーレート〉，〈フィードSW〉

文　例　TERM A, 0, 1, 1

TRON／TROFF

機　能　プログラムの実行状態を追跡する．

書　式　TRON

TROFF

文　例　TRON

WIDTH

機　能　画面に表示する文字の行数と桁数を指定する．

書　式　WIDTH〔〈桁数〉〕〔，〈行数〉〕

文　例　WIDTH 80,25

一般命令

●ステートメント

CLEAR

機　能　変数の初期化およびメモリ領域の指定をする．

書　式　CLEAR〔〈ストリング領域の大きさ〉〕，〔〈メモリの上限〉〕

文　例　CLEAR 300,&HDFFF

DATA

機　能　READ文で読み込まれる数値，文字定数を格納する．

書　式　DATA〈定数〉〔，〈定数〉…〕

文　例　DATA 1, NEC, 8001

DEF FN

機能　ユーザーによって作られた関数を定義する.

書式　DEF FN〈名前〉〔〈パラメータリスト〉〕=〈関係の定義式〉

文例　DEF FNA（X, Y）=X＊2+Y

DEFINT／DBL／STR

機能　変数の型宣言をする.

書式　DEF ｜INT｜〈文字の範囲〉
　　　　　｜SNG｜
　　　　　｜DBL｜
　　　　　｜STR｜

文例　DEFINT A, I−K

DIM

機能　配列変数の要素の大きさを指定し，メモリ領域に割り当てる.

書式　DIM〈変数名〉（〈添字の最大値〉〔,〈添字の最大値〉…〕）

文例　DIM A（12,2）

END

機能　プログラムの終了を宣言する.

書式　END

文例　END

ERASE

機能　配列変数を消去する.

書式　ERASE〈配列変数名〉〔,〈配列変数名〉…〕

文例　ERASE C, D

FOR…TO…STEP～NEXT

機能　FOR文からNEXT文までの一連の命令を指定回数だけ繰り返して実行する.

書式　FOR〈変数名〉=〈初期値〉TO〈終値〉〔STEP〈増分〉〕

NEXT〔〈変数名〉〔,〈変数名〉〕〕

文 例　FOR J=0 TO 100 STEP 2

NEXT J

GOSUB～RETURN

機 能　サブルーチンのコール，及びサブルーチンからのリターンを行う．

書 式　GOSUB〈行番号〉

RETURN

文 例　GOSUB 1000

GOTO／GO TO

機 能　指定した行へジャンプする．

書 式　1）GOTO〈行番号〉

2）GO TO〈行番号〉

文 例　GOTO 500

IF…THEN…ELSE／IF…GOTO…ELSE

機 能　論理式の条件判断を行なう．

書 式

IF〈論理式〉	THEN	〈文〉 〈行番号〉	〔ELSE	〈文〉 〈行番号〉	〕
	GOTO	〈行番号〉			

文 例　IF A$="Y" THEN 100 ELSE 120

INPUT

機 能　指定した変数へ入力する．

書 式　INPUT〔〈"プロンプト文"〉；〕〈変数〉〔,〈変数〉…〕

文 例　INPUT"Date"；A$

LET

機 能　変数に値を代入する．

書 式　〔LET〕〈変数名〉＝〈式〉

文 例　LET A=10

ＬＩＮＥ　ＩＮＰＵＴ

機　能　1行全体（255文字以内）を区切ることなく文字変数に入力する.

書　式　ＬＩＮＥ　ＩＮＰＵＴ〔〈"プロンプト文"〉；〕〈文字変数〉

文　例　ＬＩＮＥ　ＩＮＰＵＴ"ＮＡＭＥ"；ＮＡ＄

ＯＮ…ＧＯＳＵＢ／ＯＮ…ＧＯＴＯ

機　能　指定されたいつくかの行に分岐する.

書　式　ＯＮ〈式〉ＧＯＳＵＢ〈行番号〉〔，〈行番号〉…〕

　　　　ＯＮ〈式〉ＧＯＴＯ〈行番号〉〔〈行番号〉…〕

文　例　ＯＮ　Ａ　ＧＯＳＵＢ　１００，２００，３００，４００

　　　　ＯＮ　Ａ　ＧＯＴＯ　１００，２００，３００，４００

ＰＲＩＮＴ／ＬＰＲＩＮＴ

機　能　ディスプレイまたはプリンタに情報を出力する.

書　式　1）ＰＲＩＮＴ〔〈式〉…〕

　　　　2）ＬＰＲＩＮＴ〔〈式〉…〕

文　例　ＰＲＩＮＴ"ＡＢＣ"

ＰＲＩＮＴ　ＵＳＩＮＧ／ＬＰＲＩＮＴ　ＵＳＩＮＧ

機　能　文字列，数値を指定した書式で出力する.

書　式　｜ＰＲＩＮＴ　｜ＵＳＩＮＧ〈書式制御文字列〉；〈式〉〔；〈式〉…〕〔；〕
　　　　｜ＬＰＲＩＮＴ｜

文　例　ＰＲＩＮＴ　ＵＳＩＮＧ"＃＃＃＃＃，．"；Ａ，Ｂ

ＲＥＡＤ

機　能　ＤＡＴＡ文より値を読み，変数に割り当てる.

書　式　ＲＥＡＤ〈変数リスト〉

文　例　ＲＥＡＤ　Ａ，Ｂ，Ｃ＄

ＲＥＭ

機　能　プログラムに注釈を入れる.

書　式　ＲＥＭ〔〈注釈文〉〕

文 例　REM TESTPROGRAM

　　　'TESTPROGRAM

RESTORE

機 能　READ文で読むDATA文を指定する.

書 式　RESTORE〔〈行番号〉〕

文 例　RESTORE 800

STOP

機 能　プログラムの実行を停止してコマンドレベルに戻る.

書 式　STOP

文 例　STOP

SWAP

機 能　2つの変数の値を交換する.

書 式　SWAP〈変数〉,〈変数〉

文 例　SWAP A$, B$

●関数

SPC

機 能　任意の数だけ空白を出力する.

書 式　SPC(〈数式〉)

文 例　PRINT SPC(10);A$

TAB

機 能　現在のカーソルのある行の任意の位置まで空白を出力する.

書 式　TAB(〈数式〉)

文 例　PRINT TAB(10);"PC−8001"

画面・グラフィック命令

ＣＯＮＳＯＬＥ

機　能　テキスト画面のモード設定を行う．

書　式　ＣＯＮＳＯＬＥ〔〈スクロール開始行〉〕，〔〈スクロール行数〉〕
〔，〈ファンションキー表示スイッチ〉〕〔，〈カラー／白黒スイッイ〉〕

文　例　ＣＯＮＳＯＬＥ　0,25,0,1

ＬＯＣＡＴＥ

機　能　テキスト画面のカーソルの位置を指定する．

書　式　ＬＯＣＡＴＥ〔Ｘ〕〔，Ｙ〕〔，〈カーソルスイッチ〉〕

文　例　ＬＯＣＡＴＥ　10,10

ＣＳＲＬＩＮ（関数）

機　能　現在のカーソルの行位置を与える．

書　式　ＣＳＲＬＩＮ

文　例　Ｙ＝ＣＳＲＬＩＮ

ＰＯＳ（関数）

機　能　テキスト画面上の現在のカーソルの水平位置を与える．

書　式　ＰＯＳ（〈式〉）

文　例　Ｐ＝ＰＯＳ（0）

ＣＯＬＯＲ

機　能　画面に対する操作における色や機能を指定する．

書　式　ＣＯＬＯＲ〔〈ファンクションコード〉〕〔，ヌルキャラクタコード〕
〔，〈グラフィックスイッチ〉〕

文　例　ＣＯＬＯＲ　5，0，1

ＧＥＴ＠

機　能　画面上の任意の範囲のキャラクタまたは，ドットグラフィックスを配列へセーブします．

書　式　1）GET〔@〕（X_1, Y_1）－（X_2, Y_2），〈配列名〉〔, G〕

2）GET〔@〕A（X_1, Y_1）－（X_2, Y_2），〈配列名〉

文　例　GET〔@〕（50, 50）－（100, 80），A%

LINE

機　能　画面の行単位に機能を設定する，指定された2点間に線を引くなど．

書　式　1）LINE〈行指定〉，〈ファンクションコード〉

2）LINE（X_1, Y_1）－（X_2, Y_2），〈文字列〉

〔,〈ファンクションコード〉〔, | B | 〕〕
| BF |

3）LINE（X_1, Y_1）－（X_2, Y_2），〈機能〉

〔,〈ファンクションコード〉〔, | B | 〕〕
| BF |

文　例　1）LINE　7, 1

2）LINE（0, 0）－（24, 79），”X”，B

3）LINE（0, 0）－（100, 50），PSET

PRESET

機　能　画面上の任意のドットをリセットします．

書　式　PRESET（X, Y〔,〈ファンクションコード〉〕）

文　例　PRESET（100, 50）

PSET

機　能　画面上の任意のドットをセットする．

書　式　PSET（X, Y〔, ファンクションコード〕）

文　例　PSET（100, 50, 7）

PUT@

機　能　GET@文で配列へ読みこんだキャラクタやドットグラフィックスを画面の任意の位置へ表示させます．

書　式　1）PUT@（X_1, Y_1）－（X_2, Y_2），〈配列名〉，〈条件〉

2）PUT@A（X_1, Y_1）－（X_2, Y_2），〈配列名〉

文　例　　ＰＵＴ＠（５０，５０）－（１００，８０），Ａ％

ＰＯＩＮＴ（関数）

機　能　　指定した座標にドットがセットされているかいないかを与える．

書　式　　ＰＯＩＮＴ（〈Ｘ〉，〈Ｙ〉）

文　例　　ＣＬ＝ＰＯＩＮＴ（６０，６０）

算術演算命令

ＡＢＳ

機　能　　絶対値を与える．

書　式　　ＡＢＳ（〈数式〉）

文　例　　Ｂ＝ＡＢＳ（－２）

ＡＴＮ

機　能　　逆正接（アークタンジェント）を与える．

書　式　　ＡＴＮ（〈数式〉）

文　例　　Ａ＝ＡＴＮ（１）

ＣＩＮＴ

機　能　　単精度実数値，倍精度実数値を整数値に変換した値を与える．

書　式　　ＣＩＮＴ（〈数式〉）

文　例　　Ａ％＝ＣＩＮＴ（Ｂ＃＊２）

ＣＤＢＬ

機　能　　整数値，単精度実数値を倍精度実数値に変換した数値を与える．

書　式　　ＣＤＢＬ（〈数式〉）

文　例　　Ａ＃＝ＣＤＢＬ（Ｂ！／２）

ＣＯＳ

機　能　　余弦（コンサイン）を与える．

書　式　　ＣＯＳ（数式）

文　例　　C＝COS（3.1415／2）

CSNG

機　能　　整数値，倍精度実数値を単精度実数値に変換した値を与える．

書　式　　CSNG（〈数式〉）

文　例　　A！＝CSNG（B＃）

CVI／CVS／CVD（ディスク）

機　能　　文字列をデータに変換した値を与える．

書　式　　1）CVI（〈2文字の文字列〉）

　　　　　2）CVS（〈4文字の文字列〉）

　　　　　3）CVD（〈8文字の文字列〉）

文　例　　1）A％＝CVI（A＄）

　　　　　2）B＝CVS（"A3Bd"）

　　　　　3）C＃＝CVD（NUM＄）

EXP

機　能　　eに対する指数関数の値を与える．

書　式　　EXP（〈数式〉）

文　例　　E＝EXP（1）

FIX

機　能　　整数部を考える．

書　式　　FIX（〈数式〉）

文　例　　F＝FIX（B／3）

INT

機　能　　小数点以下を切り捨てた整数値を与える．

書　式　　INT（〈数式〉）

文　例　　PRINT　INT（1.123）

LOG

機　能　　自然対数を与える．

書　式　　ＬＯＧ（〈数式〉）

文　例　　ＰＲＩＮＴ　ＬＯＧ（３５／９）

ＲＮＤ

機　能　　乱数を与える．

書　式　　ＲＮＤ〔（〈機能〉）〕

文　例　　Ｒ＝ＲＮＤ（１）

ＳＧＮ

機　能　　符号を調べる．

書　式　　ＳＧＮ（〈数式〉）

文　例　　ＰＲＩＮＴ　ＳＧＮ（Ａ）

ＳＩＮ

機　能　　正弦（サイン）を与える．

書　式　　ＳＩＮ（〈数式〉）

文　例　　ＰＲＩＮＴ　ＳＩＮ（１）

ＳＱＲ

機　能　　平方根（ルート）を与える．

書　式　　ＳＱＲ（〈数式〉）

文　例　　Ａ＝ＳＱＲ（２）

ＴＡＮ

機　能　　正接（タンジェント）を与える．

書　式　　ＴＡＮ（〈数式〉）

文　例　　Ａ＝ＴＡＮ（１．２３）

文字操作命令

●ステートメント

ＭＩＤ＄

機　能　　文字列の一部を置き替える．

書　式　　ＭＩＤ＄（〈文字列１〉，〈式１〉〔，〈式２〉〕＝〈文字列２〉）

文　例　　ＭＩＤ＄（Ａ＄，３）＝“ＡＢＣ”

●関数

ＡＳＣ

機　能　　文字のキャラクタコードを与える．

書　式　　ＡＳＣ（〈文字列〉）

文　例　　Ａ＝ＡＳＣ（“Ａ”）

ＣＨＲ＄

機　能　　指定したキャラクタコードを持つ文字を与える．

書　式　　ＣＨＲ＄（〈数式〉）

文　例　　Ａ＄＝ＣＨＲ＄（６５）

ＨＥＸ＄

機　能　　１０進数を１６進数に変換し，その文字列を与える．

書　式　　ＨＥＸ＄（〈数式〉）

文　例　　Ａ＄＝ＨＥＸ＄（Ｘ）

ＩＮＫＥＹ＄

機　能　　キーが押されていれば，その文字を，キーが押されなければヌルストリングを与える．

書　式　　ＩＮＫＥＹ＄

文　例　　Ａ＄＝ＩＮＫＥＹ＄

ＩＮＳＴＲ

機 能　文字列の中から任意の文字列を捜してその文字の位置を与える.

書 式　ＩＮＳＴＲ（〔〈数式〉，〕〈文字列１〉，〈文字列２〉）

文 例　Ｊ＝ＩＮＳＴＲ（Ａ＄，“Ｊ”）

ＬＥＦＴ＄

機 能　文字列の左側から任意の長さの文字列を与える.

書 式　ＬＥＦＴ＄（〈文字列〉，〈数式〉）

文 例　Ｂ＄＝ＬＥＦＴ＄（Ａ＄，４）

ＬＥＮ

機 能　文字列の総文字数を与える.

書 式　ＬＥＮ（〈文字列〉）

文 例　ＰＲＩＮＴ　ＬＥＮ（Ａ＄）

ＭＩＤ＄

機 能　文字列の中から任意の長さの文字列を与える.

書 式　ＭＩＤ＄（〈文字列〉，〈式１〉〔，〈式２〉〕）

文 例　ＰＲＩＮＴ　ＭＩＤ＄（Ａ＄，2,3）

ＭＫＩ＄／ＭＫＳ＄／ＭＫＤ＄（ディスク）

機 能　各数値を内部表現に対応した文字コードに変換する.

書 式　１）ＭＫＩ＄（〈整数表記〉）

２）ＭＫＳ＄（〈単精度表記〉）

３）ＭＫＤ＄（〈倍精度表記〉）

文 例　１）Ａ＄＝ＭＫＩ＄（１２８）

２）Ｂ＄＝ＭＫＳ＄（1.23）

３）Ｃ＄＝ＭＫＤ＄（3.141592654＃）

ＯＣＴ＄

機 能　１０進数を８進数に変換し，その文字列を与える.

書 式　ＯＣＴ＄（〈数式〉）

文　例　　ＰＲＩＮＴ　ＯＣＴ＄（Ａ）

ＲＩＧＨＴ＄

機　能　　文字列の右側から任意の長さの文字列を与える．

書　式　　ＲＩＧＨＴ＄（〈文字列〉，〈数式〉）

文　例　　ＰＲＩＮＴ　ＲＩＧＨＴ＄（Ａ＄，４）

ＳＰＡＣＥ＄

機　能　　任意の長さの空白文字を与える．

書　式　　ＳＰＡＣＥ＄（〈数式〉）

文　例　　ＰＲＩＮＴ　ＳＰＡＣＥ＄（１０）；"ＡＢＣ"

ＳＴＲ＄

機　能　　数値を表す文字列を与える．

書　式　　ＳＴＲ＄（〈数式〉）

文　例　　Ａ＄＝ＳＴＲ＄（１２３）

ＳＴＲＩＮＧ＄

機　能　　任意の文字を任意の数だけ与える．

書　式　　ＳＴＲＩＮＧ＄（〈式〉，〈文字式〉）

文　例　　Ａ＄＝ＳＴＲＩＮＧ＄（10，"＋"）

ＶＡＬ

機　能　　文字列の表す数値を与える．

書　式　　ＶＡＬ（〈文字列〉）

文　例　　Ａ＝ＶＡＬ（"２"）

入出力命令

●ステートメント

ＣＬＯＳＥ（ディスク）

機　能　　ファイルを閉じる．

書　式　CLOSE〔〔#〕〈ファイル番号〉〔,〔#〕〈ファイル番号〉〕…〕

文　例　CLOSE

FIELD（ディスク）

機　能　ランダムファイル・バッファに変数領域を割り当てる.

書　式　FIELD〔#〕〈ファイル番号〉,〈フィールド幅〉AS〈文字変数〉
〔,〈フィールド幅〉AS〈文字変数〉…〕

文　例　FIELD#1, 128 AS A$, 64 AS B$

GET（ディスク）

機　能　ファイル中のデータをバッファに読み込む.

書　式　GET〔#〕〈ファイル番号〉〔,〈数〉〕

文　例　GET 3, 5

INPUT#

機　能　シーケンシャルファイルからのデータを読み込む.

書　式　INPUT#〈ファイル番号〉,〈変数〉〔,〈変数〉…〕

文　例　INPUT#1, A, B

LINE INPUT#（ディスク）

機　能　1行全体（254文字以内）を区切ることなくシーケンシャル・ディスクファイルより文字変数に読み込む.

書　式　LINE INPUT#〈ファイル番号〉,〈文字変数〉

文　例　LINE INPUT #1, A$

LSET／RSET（ディスク）

機　能　ファイルバッファにデータを書き込む.

書　式　LSET〈文字型変数〉=〈文字列〉
RSET〈文字型変数〉=〈文字列〉

文　例　LSET A$="PC−8001"
RSET A$="N−BASIC"

ＯＰＥＮ（ディスク）

機　能　ファイルを開く．

書　式　ＯＰＥＮ”〈ファイル名〉”〔ＦＯＲ〈モード〉〕ＡＳ〔＃〕〈ファイル番号〉

文　例　ＯＰＥＮ”ＴＥＳＴ”ＦＯＲ　ＩＮＰＵＴ　ＡＳ＃１

ＰＲＩＮＴ＃

機　能　シーケンシャルファイルにデータを出力する．

書　式　ＰＲＩＮＴ　＃〈ファイル番号〉，〔〈式〉…〕

文　例　ＰＲＩＮＴ＃２，Ａ，Ｂ，Ｃ

ＰＲＩＮＴ＃　ＵＳＩＮＧ（ディスク）

機　能　文字列，数値を指定した書式でファイルに出力する．

書　式　ＰＲＩＮＴ＃〈ファイル番号〉，ＵＳＩＮＧ〈書式制御文字列〉；〈式〉〔；〈式〉…〕〔；〕

文　例　ＰＲＩＮＴ＃２，ＵＳＩＮＧ“＃＃＃＃”；Ａ

ＰＵＴ（ディスク）

機　能　バッファ中のデータをファイルに書き出す．

書　式　ＰＵＴ〔＃〕〈ファイル番号〉〔，〈数〉〕

文　例　ＰＵＴ＃３，５

●関数

ＥＯＦ（ディスク）

機　能　ファイルの終了コードを与える．

書　式　ＥＯＦ（〈ファイル番号〉）

文　例　ＩＦ　ＥＯＦ（１）ＴＨＥＮ　ＣＬＯＳＥ　１

ＦＰＯＳ（ディスク）

機　能　ファイル中でのヘッドの物理的な現在位置を与える．

書　式　ＦＰＯＳ（〈ファイル番号〉）

文 例　HEAD＝FPOS（4）

INPUT$（ディスク）

機 能　指定されたファイルより指定された長さの文字を与える．

書 式　INPUT$（<文字数>〔,〔#〕<ファイル番号>〕）

文 例　WORD$＝INPUT$（6，#2）

LOC（ディスク）

機 能　ファイル中での論理的な現在位置を与える．

書 式　LOC（ファイル番号）

文 例　LAST＝LOC（2）

LOF（ディスク）

機 能　ファイルの大きさを与える．

書 式　LOF（<ファイル番号>）

文 例　MAXREC＝LOF（2）

キー制御命令

●ステートメント

KEY

機 能　キーボードの上部にあるプログラマブル・ファンクションキーの内容を定義する．

書 式　KEY <キー番号>，<文字列>

文 例　KEY1，"LOAD"

クロック制御命令

●関数

DATE$

機　能　日付を与える.

書　式　DATE$

文　例　D$=DATE$

TIME$

機　能　内蔵クロックの時刻を与える.

書　式　1）TIME$

　　　　2）TIME$="HH：MM：SS"

文　例　PRINT　TIME$

プリンタ制御命令

●関数

LPOS

機　能　現在のプリンタのヘッド位置を与えます.

書　式　LPOS（<式>）

文　例　LPOS（0）

ディスク制御命令

●ステートメント

DSKO$（ディスク）

機 能　ディスクに対しての直接書き込み.

書 式　1）DSKO$〈ドライブ番号〉，〈トラック番号〉，〈セクタ番号〉

2）DSKO$〈ドライブ番号〉，〈サーフェス番号〉，〈トラック番号〉，〈セクタ番号〉

文 例　1）DSKO$　2,10,1

2）DSKO$　1,0,19,1

KILL（ディスク）

機 能　ディスクファイルを削除する.

書 式　KILL〈ファイル名〉

文 例　KILL"2：SAMPLE"

NAME（ディスク）

機 能　ディスクファイルの名前を変える.

書 式　NAME〈旧ファイル名〉AS〈新ファイル名〉

文 例　NAME"SAMPLE"AS"SAMPLEI"

SET（ディスク）

機 能　ファイル属性のセット・リセットを行なう.

書 式　1）SET〈ドライブ番号〉，〈"属性文字"〉

2）SET〈ファイル名〉，〈"属性文字"〉

3）SET#〈ファイル番号〉，〈"属性文字"〉

文 例　1）SET　1，"P"

2）SET"2：PROTEC"，"P"

3）SET#1，"R"

●関数

ＡＴＴＲ＄（ディスク）

機　能　　指定ファイル，ドライブの属性を与える．

書　式　　ＡＴＴＲ＄｜（〈ドライブ番号〉）｜
　　　　　　　　　　｜（＃〈ファイル番号〉）｜
　　　　　　　　　　｜（〈ファイル名〉）｜

文　例　　ＰＲＩＮＴ　ＡＴＴＲ＄（1）

ＤＳＫＦ（ディスク）

機　能　　ディスクの諸元を与える．

書　式　　ＤＳＫＦ（〈ドライブ番号〉）

文　例　　ＰＲＩＮＴ　ＤＳＫＦ（1）

ＤＳＫＩ＄（ディスク）

機　能　　ディスクから直接読み出した内容を文字列として与える．

書　式　　1）ＤＳＫＩ＄（〈ドライブ番号〉，〈トラック番号〉，〈セクタ番号〉）
　　　　　2）ＤＳＫＩ＄（〈ドライブ番号〉，〈サーフェス番号〉，〈トラック番号〉，〈セクタ番号〉）

文　例　　1）Ｄ＄＝ＤＳＫＩ＄（2，19，1）
　　　　　2）Ｄ＄＝ＤＳＫＩ＄（2，1，19，1）

特殊命令

●ステートメント

ＢＥＥＰ

機　能　　内蔵スピーカを鳴らす．

書　式　　ＢＥＥＰ〔〈スイッチ〉〕

文　例　　ＢＥＥＰ

DEF USR

機 能　機械語を作られたユーザー関数の実行開始番地を定義する.

書 式　DEF USR〔〈番号〉〕＝〈開始番地〉

文 例　DEF USR3＝&HE000

ERROR

機 能　エラー発生のシミュレート，エラー番号のユーザー定義をする.

書 式　ERROR〈整数表記〉

文 例　ERROR 200

MOTOR

機 能　カセットテープレコーダのモータのON，OFFを制御する.

書 式　MOTOR〔〈スイッチ〉〕

文 例　MOTOR 1

ON ERROR GOTO

機 能　エラートラップを可能にし，エラー処理ルーチンの開始行を定義する.

書 式　ON ERROR GOTO〈行番号〉

文 例　ON ERROR GOTO 1000

OUT

機 能　コンピュータの出力ポートに1バイトのデータを送る.

書 式　OUT〈ポートアドレス〉，〈式〉

文 例　OUT 80，0

POKE

機 能　メモリ上の指定番地へデータを書き込む.

書 式　POKE〈番地〉，〈整数表記〉

文 例　POKE &HE000，&HFF

RESUME

機 能　エラー回復処理終了後，プログラムの実行を再開する.

書　式　RESUME　| 〔0〕 / NEXT / 〈行番号〉 |

文　例　RESUME　0

RESUME　NEXT

RESUME　1000

WAIT

機　能　コンピュータの入力ポートをモニタする間，プログラムの実行を停止する．

書　式　WAIT〈ポート番号〉，〈式1〉〔，〈式2〉〕

文　例　WAIT　1,&H22,&H22

●関数

ERL／ERR

機　能　発生したエラーのエラーコード及びエラーの発生した行番号を与える．

書　式　ERL

ERR

文　例　ER＝ERL

EL＝ERR

FRE

機　能　メモリの未使用領域の大きさを与える．

書　式　FRE（〈機能〉）

文　例　PRINT　FRE（0）

INP

機　能　入力ポートから値を得る．

書　式　INP（〈ポート番号〉）

文　例　A＝INP（15）

PEEK

機　能　メモリ上の指定された番地の内容を読み出す．

書　式　ＰＥＥＫ（〈番地〉）

文　例　Ａ＝ＰＥＥＫ（&ＨＥ０００）

ＵＳＲ

機　能　機械語で作られた関数ルーチンを呼び出す．

書　式　ＵＳＲ〔〈番号〉〕（〈引数〉）

文　例　Ｉ＝ＵＳＲ３（Ｊ）

ＶＡＲＰＴＲ

機　能　変数の格納されているメモリの番地，ファイルに割り当てられているバッファの開始番地を与える．

書　式　１）ＶＡＲＰＴＲ（〈変数名〉）

　　　　２）ＶＡＲＰＴＲ（＃〈ファイル番号〉）

文　例　１）ＰＲＩＮＴ　ＨＥＸ＄（ＶＡＲＰＴＲ（Ａ））

　　　　２）ＢＵＡＤ＝ＶＡＲＰＴＲ（＃１）

キャラクタコード表

上位4ビット→

下位4ビット↓

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		D_E		0	@	P		p				ー	タ	ミ		
1	S_H	D_1	!	1	A	Q	a	q			。	ア	チ	ム		円
2	S_X	D_2	"	2	B	R	b	r			「	イ	ツ	メ		年
3	E_X	D_3	#	3	C	S	c	s			」	ウ	テ	モ		月
4	E_T	D_4	$	4	D	T	d	t			、	エ	ト	ヤ		日
5	E_Q	N_K	%	5	E	U	e	u			・	オ	ナ	ユ		時
6	A_K	S_N	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		分
7	B_L	E_B	'	7	G	W	g	w			ァ	キ	ヌ	ラ		秒
8	B_S	C_N	(	8	H	X	h	x			ィ	ク	ネ	リ	♠	
9	H_T	E_M	)	9	I	Y	i	y			ゥ	ケ	ノ	ル	♥	
A	L_F	S_B	*	:	J	Z	j	z			ェ	コ	ハ	レ	♦	
B	H_M	E_C	+	;	K	[	k	{			ォ	サ	ヒ	ロ	♣	
C	C_L	→	,	<	L	¥	l	¦			ャ	シ	フ	ワ	●	
D	C_R	←	−	=	M	]	m	}			ュ	ス	ヘ	ン	○	
E	S_O	↑	.	>	N	^	n	~			ョ	セ	ホ	゛		
F	S_I	↓	/	?	O	_	o				ッ	ソ	マ	゜		

5.3

Z80
インストラクション活用表

Z80 インストラクション・セット

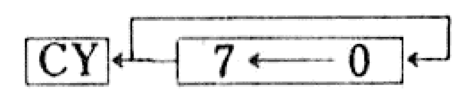
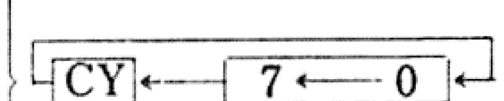
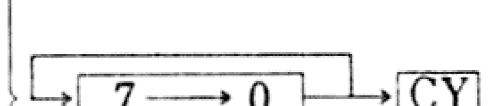
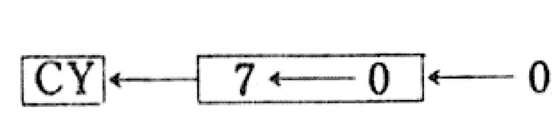
命令群	ニーモニック	オペレーション	フラグ S Z H P/V N C	バイト	ステート	OPコード 76 543 210
8ビット・ロード命令	LD r, r′	r←r′	• • • • • • •	1	4	01 r r′ Ⓔ
	LD r, n	r←n	• • • • • • •	2	7	00 r 110 Ⓔ ← n →
	LD r, (HL)	r←(HL)	• • • • • • •	1	7	01 r 110 Ⓔ
	LD r, (IX+d)	r←(IX+d)	• • • • • • •	3	19	11 011 101 01 r 110 Ⓔ ← d →
	LD r, (IY+d)	r←(IY+d)	• • • • • • •	3	19	11 111 101 01 r 110 Ⓔ ← d →
	LD (HL), r	(HL)←r	• • • • • • •	1	7	01 110 r Ⓔ
	LD (IX+d), r	(IX+d)←r	• • • • • • •	3	19	11 011 101 01 110 r Ⓔ ← d →
	LD (IY+d), r	(IY+d)←r	• • • • • • •	3	19	11 111 101 01 110 r Ⓔ ← d →
	LD (HL), n	(HL)←n	• • • • • • •	2	10	00 110 110 ← n →
	LD (IX+d), n	(IX+d)←n	• • • • • • •	4	19	11 011 101 00 110 110 ← d → ← n →
	LD (IY+d), n	(IY+d)←n	• • • • • • •	4	19	11 111 101 00 110 110 ← d → ← n →
	LD A, (BC)	A←(BC)	• • • • • • •	1	7	00 001 010
	LD A, (DE)	A←(DE)	• • • • • • •	1	7	00 011 010
	LD A, (nn)	A←(nn)	• • • • • • •	3	13	00 111 010 ← n → ← n →
	LD (BC), A	(BC)←A	• • • • • • •	1	7	00 000 010
	LD (DE), A	(DE)←A	• • • • • • •	1	7	00 010 010
	LD (nn), A	(nn)←A	• • • • • • •	3	13	00 110 010 ← n → ← n →
	LD A, I	A←I	↕ ↕ 0 IFF 0 •	2	9	11 101 101 01 010 111
	LD A, R	A←R	↕ ↕ 0 IFF 0 •	2	9	11 101 101 01 011 111
	LD I, A	I←A	• • • • • • •	2	9	11 101 101 01 000 111
	LD R, A	R←A	• • • • • • •	2	9	11 101 101 01 001 111
16ビット・ロード命令	LD dd, nn	dd←nn	• • • • • • •	3	10	00 dd0 001 Ⓐ ← n → ← n →
	LD IX, nn	IX←nn	• • • • • • •	4	14	11 011 101 00 100 001 ← n → ← n →
	LD IY, nn	IY←nn	• • • • • • •	4	14	11 111 101 00 100 001 ← n → ← n →
	LD HL, (nn)	H←(nn+1) L←(nn)	• • • • • • •	3	16	00 101 010 ← n → ← n →
	LD dd, (nn)	dd_H←(nn+1) dd_L←(nn)	• • • • • • •	4	20	11 101 101 01 dd1 011 Ⓐ ← n → ← n →

命令群	ニーモニック	オペレーション	フラグ S Z H P/V N C	バイト	ステート	OPコード 76 543 210	
16ビット・ロード命令	LD IX, (nn)	IX_H←(nn+1) IX_L←(nn)	• • • • • • •	4	20	11 011 101 00 101 010 ← n → ← n →	
	LD IY, (nn)	IY_H←(nn+1) IY_L←(nn)	• • • • • • •	4	20	11 111 101 00 101 010 ← n → ← n →	
	LD (nn), HL	(nn+1)←H (nn)←L	• • • • • • •	3	16	00 100 010 ← n → ← n →	
	LD (nn), dd	(nn+1)←dd_H (nn)←dd_L	• • • • • • •	4	20	11 101 101 01 dd0 011 ← n → ← n →	Ⓐ
	LD (nn), IX	(nn+1)←IX_H (nn)←IX_L	• • • • • • •	4	20	11 011 101 00 100 010 ← n → ← n →	
	LD (nn), IY	(nn+1)←IY_H (nn)←IY_L	• • • • • • •	4	20	11 111 101 00 100 010 ← n → ← n →	
	LD SP, HL	SP←HL	• • • • • • •	1	6	11 111 001	
	LD SP, IX	SP←IX	• • • • • • •	2	10	11 011 101 11 111 001	
	LD SP, IY	SP←IY	• • • • • • •	2	10	11 111 101 11 111 001	
	PUSH qq	(SP−2)←qq_L (SP−1)←qq_H	• • • • • • •	1	11	11 qq0 101	Ⓑ
	PUSH IX	(SP−2)←IX_L (SP−1)←IX_H	• • • • • • •	2	15	11 011 101 11 100 101	
	PUSH IY	(SP−2)←IY_L (SP−1)←IY_H	• • • • • • •	2	15	11 111 101 11 100 101	
	POP qq	qq_H←(SP+1) qq_L←(SP)	• • • • • • •	1	10	11 qq0 001	Ⓑ
	POP IX	IX_H←(SP+1) IX_L←(SP)	• • • • • • •	2	14	11 011 101 11 100 001	
	POP IY	IY_H←(SP+1) IY_L←(SP)	• • • • • • •	2	14	11 111 101 11 100 001	
エクスチェンジ命令	EX DE, HL	DE↔HL	• • • • • • •	1	4	11 101 011	
	EX AF, AF′	AF↔AF′	• • • • • • •	1	4	00 001 000	
	EXX	(BC↔BC′ DE↔DE′ HL↔HL′)	• • • • • • •	1	4	11 011 001	
	EX (SP), HL	H↔(SP+1) L↔(SP)	• • • • • • •	1	19	11 100 011	
	EX (SP), IX	IX_H↔(SP+1) IX_L↔(SP)	• • • • • • •	2	23	11 011 101 11 100 011	
	EX (SP), IY	IY_H↔(SP+1) IY_L↔(SP)	• • • • • • •	2	23	11 111 101 11 100 011	
ブロック転送命令	LDI	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1	• • 0 ↕① 0 •	2	16	11 101 101 10 100 000	
	LDIR	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1 until BC=0	• • 0 0 0 •	2	21 if BC≠0 16 if BC=0	11 101 101 10 110 000	

命令群	ニーモニック	オペレーション	フラグ S Z H P/V N C	バイト	ステート	OPコード 76 543 210
ブロック転送命令	**LDD**	(DE)←(HL) DE←DE−1 HL←HL−1 BC←BC−1	• • 0 ↕① 0 •	2	16	11 101 101 10 101 000
	LDDR	(DE)←(HL) DE←DE−1 HL←HL−1 BC←BC−1 until BC=0	• • 0 0 0 •	2	21 if BC≠0 16 if BC=0	11 101 101 10 111 000
ブロック・サーチ命令	CPI	A−(HL) HL←HL+1 BC←BC−1	↕ ↕② ↕ ↕① 1 •	2	16	11 101 101 10 100 001
	CPIR	A−(HL) HL←HL+1 BC←BC−1 until A=(HL) or BC=0	↕ ↕② ↕ ↕① 1 •	2	21 if BC≠0 and A≠(HL) 16 if BC=0 or A=(HL)	11 101 101 10 110 001
	CPD	A−(HL) HL←HL−1 BC←BC−1	↕ ↕② ↕ ↕① 1 •	2	16	11 101 101 10 101 001
	CPDR	A−(HL) HL←HL−1 BC←BC−1 until A=(HL) or BC=0	↕ ↕② ↕ ↕① 1 •	2	21 if BC≠0 and A≠(HL) 16 if BC=0 or A=(HL)	11 101 101 10 111 001
8ビット算術論理演算命令	ADD A, r	A←A+r	↕ ↕ ↕ V 0 ↕	1	4	10 000 r Ⓔ
	ADD A, n	A←A+n	↕ ↕ ↕ V 0 ↕	2	7	11 000 110 ← n →
	ADD A, (HL)	A←A+(HL)	↕ ↕ ↕ V 0 ↕	1	7	10 000 110
	ADD A, (IX+d)	A←A+(IX+d)	↕ ↕ ↕ V 0 ↕	3	19	11 011 101 10 000 110 ← d →
	ADD, A, (IY+d)	A←A+(IY+d)	↕ ↕ ↕ V 0 ↕	3	19	11 111 101 10 000 110 ← d →
	ADC A, r	A←A+r+CY	↕ ↕ ↕ V 0 ↕	1	4	10 001 r Ⓔ
	ADC A, n	A←A+n+CY	↕ ↕ ↕ V 0 ↕	2	7	11 001 110 ← n →
	ADC A, (HL)	A←A+(HL)+CY	↕ ↕ ↕ V 0 ↕	1	7	10 001 110
	ADC A, (IX+d)	A←A+(IX+d)+CY	↕ ↕ ↕ V 0 ↕	3	19	11 011 101 10 001 110 ← d →
	ADC A, (IY+d)	A←A+(IY+d)+CY	↕ ↕ ↕ V 0 ↕	3	19	11 111 101 10 001 110 ← d →
	SUB r	A←A−r	↕ ↕ ↕ V 1 ↕	1	4	10 010 r Ⓔ
	SUB n	A←A−n	↕ ↕ ↕ V 1 ↕	2	7	11 010 110 ← n →
	SUB (HL)	A←A−(HL)	↕ ↕ ↕ V 1 ↕	1	7	10 010 110
	SUB (IX+d)	A←A−(IX+d)	↕ ↕ ↕ V 1 ↕	3	19	11 011 101 10 010 110 ← d →
	SUB (IY+d)	A←A−(IY+d)	↕ ↕ ↕ V 1 ↕	3	19	11 111 101 10 010 110 ← d →
	SBC A, r	A←A−r−CY	↕ ↕ ↕ V 1 ↕	1	4	10 011 r Ⓔ
	SBC A, n	A←A−n−CY	↕ ↕ ↕ V 1 ↕	2	7	11 011 110 ← n →
	SBC A, (HL)	A←A−(HL)−CY	↕ ↕ ↕ V 1 ↕	1	7	10 011 110

命令群	ニーモニック	オペレーション	フラグ S	Z	H	P/V	N	C	バイト	ステート	OPコード 76 543 210
8ビット算術論理演算命令	SBC A, (IX+d)	A←A−(IX+d)−CY	↕	↕	↕	V	1	↕	3	19	11 011 101 10 011 110 ← d →
	SBC A, (IY+d)	A←A−(IY+d)−CY	↕	↕	↕	V	1	↕	3	19	11 111 101 10 011 110 ← d →
	AND r	A←A∧r	↕	↕	1	P	0	0	1	4	10 100 r Ⓔ
	AND n	A←A∧n	↕	↕	1	P	0	0	2	7	11 100 110 ← n →
	AND (HL)	A←A∧(HL)	↕	↕	1	P	0	0	1	7	10 100 110
	AND (IX+d)	A←A∧(IX+d)	↕	↕	1	P	0	0	3	19	11 011 101 10 100 110 ← d →
	AND (IY+d)	A←A∧(IY+d)	↕	↕	1	P	0	0	3	19	11 111 101 10 100 110 ← d →
	OR r	A←A∨r	↕	↕	0	P	0	0	1	4	10 110 r Ⓔ
	OR n	A←A∨n	↕	↕	0	P	0	0	2	7	11 110 110 ← n →
	OR (HL)	A←A∨(HL)	↕	↕	0	P	0	0	1	7	10 110 110
	OR (IX+d)	A←A∨(IX+d)	↕	↕	0	P	0	0	3	19	11 011 101 10 110 110 ← d →
	OR (IY+d)	A←A∨(IY+d)	↕	↕	0	P	0	0	3	19	11 111 101 10 110 110 ← d →
	XOR r	A←A∀r	↕	↕	0	P	0	0	1	4	10 101 r Ⓔ
	XOR n	A←A∀n	↕	↕	0	P	0	0	2	7	11 101 110 ← n →
	XOR (HL)	A←A∀(HL)	↕	↕	0	P	0	0	1	7	10 101 110
	XOR (IX+d)	A←A∀(IX+d)	↕	↕	0	P	0	0	3	19	11 011 101 10 101 110 ← d →
	XOR (IY+d)	A←A∀(IY+d)	↕	↕	0	P	0	0	3	19	11 111 101 10 101 110 ← d →
	CP r	A−r	↕	↕	↕	V	1	↕	1	4	10 111 r Ⓔ
	CP n	A−n	↕	↕	↕	V	1	↕	2	7	11 111 110 ← n →
	CP (HL)	A−(HL)	↕	↕	↕	V	1	↕	1	7	10 111 110
	CP (IX+d)	A−(IX+d)	↕	↕	↕	V	1	↕	3	19	11 011 101 10 111 110 ← d →
	CP (IY+d)	A−(IY+d)	↕	↕	↕	V	1	↕	3	19	11 111 101 10 111 110 ← d →
	INC r	r←r+1	↕	↕	↕	V	0	•	1	4	00 r 100 Ⓔ
	INC (HL)	(HL)←(HL)+1	↕	↕	↕	V	0	•	1	11	00 110 100
	INC (IX+d)	(IX+d)←(IX+d)+1	↕	↕	↕	V	0	•	3	23	11 011 101 00 110 100 ← d →
	INC (IY+d)	(IY+d)←(IX+d)+1	↕	↕	↕	V	0	•	3	23	11 111 101 00 110 100 ← d →
	DEC r	r←r−1	↕	↕	↕	V	1	•	1	4	00 r 101 Ⓔ
	DEC (HL)	(HL)←(HL)−1	↕	↕	↕	V	1	•	1	11	00 110 101
	DEC (IX+d)	(IX+d)←(IX+d)−1	↕	↕	↕	V	1	•	3	23	11 011 101 00 110 101 ← d →

命令群	ニーモニック	オペレーション	フラグ S	Z	H	P/V	N	C	バイト	ステート	OPコード 76 543 210
8ビット算術論理演算命令	DEC (IY+d)	(IY+d)←(IY+d)−1	↕	↕	↕	V	1	•	3	23	11 111 101 00 110 101 ← d →
16ビット算術演算命令	ADD HL, ss	HL←HL+ss	•	•	×	•	0	↕	1	11	00 ss1 001 Ⓐ
	ADC HL, ss	HL←HL+ss+CY	↕	↕	×	V	0	↕	2	15	11 101 101 01 ss1 010 Ⓐ
	SBC HL, ss	HL←HL−ss−CY	↕	↕	×	V	1	↕	2	15	11 101 101 01 ss0 010 Ⓐ
	ADD IX, pp	IX←IX+pp	•	•	×	•	0	↕	2	15	11 011 101 00 pp1 001 Ⓒ
	ADD IY, rr	IY←IY+rr	•	•	×	•	0	↕	2	15	11 111 101 00 rr1 001 Ⓓ
	INC ss	ss←ss+1	•	•	•	•	•	•	1	6	00 ss0 011 Ⓐ
	INC IX	IX←IX+1	•	•	•	•	•	•	2	10	11 011 101 00 100 011
	INC IY	IY←IY+1	•	•	•	•	•	•	2	10	11 111 101 00 100 011
	DEC ss	ss←ss−1	•	•	•	•	•	•	1	6	00 ss1 011 Ⓐ
	DEC IX	IX←IX−1	•	•	•	•	•	•	2	10	11 011 101 00 101 011
	DEC IY	IY←IY−1	•	•	•	•	•	•	2	10	11 111 101 00 101 011
アキュムレータ操作命令	DAA	Decimal adjust Acc	↕	↕	↕	P	•	↕	1	4	00 100 111
	CPL	A←$\bar{A}$	•	•	1	•	1	•	1	4	00 101 111
	NEG	A←$\bar{A}$+1	↕	↕	↕	V	1	↕	2	8	11 101 101 01 000 100
	CCF	CY←$\overline{CY}$	•	•	×	•	0	↕	1	4	00 111 111
	SCF	CY←1	•	•	0	•	0	1	1	4	00 110 111
CPUコントロール命令	NOP	No operation	•	•	•	•	•	•	1	4	00 000 000
	HALT	CPU halted	•	•	•	•	•	•	1	4	01 110 110
	DI	IFF←0	•	•	•	•	•	•	1	4	11 110 011
	EI	IFF←1	•	•	•	•	•	•	1	4	11 111 011
	IM 0	Set interrupt mode 0	•	•	•	•	•	•	2	8	11 101 101 01 000 110
	IM 1	Set interrupt mode 1	•	•	•	•	•	•	2	8	11 101 101 01 010 110
	IM 2	Set interrupt mode 2	•	•	•	•	•	•	2	8	11 101 101 01 011 110
ローテート・シフト命令	RLCA	CY ← [7 ← 0] ← (bit 7 → bit 0) A	•	•	0	•	0	↕	1	4	00 000 111
	RLA	┌→ CY ← [7 ← 0] ←┐ (CY → bit 0) A	•	•	0	•	0	↕	1	4	00 010 111
	RRCA	(bit 0 → bit 7) [7 → 0] → CY A	•	•	0	•	0	↕	1	4	00 001 111
	RRA	┌→ [7 → 0] → CY ┐ (CY → bit 7) A	•	•	0	•	0	↕	1	4	00 011 111

命令群	ニーモニック	オペレーション	S	Z	H	P/V	N	C	バイト	ステート	OPコード 76 543 210
ローテート・シフト命令	RLC r		↕	↕	0	P	0	↕	2	8	11 001 011 00 000 r Ⓔ
	RLC (HL)		↕	↕	0	P	0	↕	2	15	11 001 011 00 000 110
	RLC (IX+d)	 r, (HL), (IX+d), (IY+d)	↕	↕	0	P	0	↕	4	23	11 011 101 11 001 011 ← d → 00 000 110
	RLC (IY+d)		↕	↕	0	P	0	↕	4	23	11 111 101 11 001 011 ← d → 00 000 110
	RL r		↕	↕	0	P	0	↕	2	8	11 001 011 00 010 r Ⓔ
	RL (HL)		↕	↕	0	P	0	↕	2	15	11 001 011 00 010 110
	RL (IX+d)	 r, (HL), (IX+d), (IY+d)	↕	↕	0	P	0	↕	4	23	11 011 101 11 001 011 ← d → 00 010 110
	RL (IY+d)		↕	↕	0	P	0	↕	4	23	11 111 101 11 001 011 ← d → 00 010 110
	RRC r		↕	↕	0	P	0	↕	2	8	11 001 011 00 001 r Ⓔ
	RRC (HL)		↕	↕	0	P	0	↕	2	15	11 001 011 00 001 110
	RRC (IX+d)	 r, (HL), (IX+d), (IY+d)	↕	↕	0	P	0	↕	4	23	11 011 101 11 001 011 ← d → 00 001 110
	RRC (IY+d)		↕	↕	0	P	0	↕	4	23	11 111 101 11 001 011 ← d → 00 001 110
	RR r		↕	↕	0	P	0	↕	2	8	11 001 011 00 011 r Ⓔ
	RR (HL)		↕	↕	0	P	0	↕	2	15	11 001 011 00 011 110
	RR (IX+d)	 r, (HL), (IX+d), (IY+d)	↕	↕	0	P	0	↕	4	23	11 011 101 11 001 011 ← d → 00 011 110
	RR (IY+d)		↕	↕	0	P	0	↕	4	23	11 111 101 11 001 011 ← d → 00 011 110
	SLA r		↕	↕	0	P	0	↕	2	8	11 001 011 00 100 r Ⓔ
	SLA (HL)		↕	↕	0	P	0	↕	2	15	11 001 011 00 100 110
	SLA (IX+d)	 r, (HL), (IX+d), (IY+d)	↕	↕	0	P	0	↕	4	23	11 011 101 11 001 011 ← d → 00 100 110
	SLA (IY+d)		↕	↕	0	P	0	↕	4	23	11 111 101 11 001 011 ← d → 00 100 110

命令群	ニーモニック	オペレーション	S	Z	H	P/V	N	C	バイト	ステート	OPコード 76 543 210
ローテート・シフト命令	SRA r		↕	↕	0	P	0	↕	2	8	11 001 011 00 101 r Ⓔ
	SRA (HL)		↕	↕	0	P	0	↕	2	15	11 001 011 00 101 110
	SRA (IX+d)	┌[7 ⟶ 0]⟶[CY] (bit 7 fed back into bit 7) r, (HL), (IX+d), (IY+d)	↕	↕	0	P	0	↕	4	23	11 011 101 11 001 011 ← d → 00 101 110
	SRA (IY+d)		↕	↕	0	P	0	↕	4	23	11 111 101 11 001 011 ← d → 00 101 110
	SRL r		↕	↕	0	P	0	↕	2	8	11 001 011 00 111 r Ⓔ
	SRL (HL)		↕	↕	0	P	0	↕	2	15	11 001 011 00 111 110
	SRL (IX+d)	0→[7 ⟶ 0]⟶[CY] r, (HL), (IX+d), (IY+d)	↕	↕	0	P	0	↕	4	23	11 011 101 11 001 011 ← d → 00 111 110
	SRL (IY+d)		↕	↕	0	P	0	↕	4	23	11 111 101 11 001 011 ← d → 00 111 110
	RLD	A [7−4|3−0] [7−4|3−0](HL)	↕	↕	0	P	0	•	2	18	11 101 101 01 101 111
	RRD	A [7−4|3−0] [7−4|3−0](HL)	↕	↕	0	P	0	•	2	18	11 101 101 01 100 111
ビット操作命令	BIT b, r	$Z \leftarrow \overline{r}_b$	×	↕	1	×	0	•	2	8	11 001 011 01 b r ⒺⒻ
	BIT b, (HL)	$Z \leftarrow \overline{(HL)}_b$	×	↕	1	×	0	•	2	12	11 001 011 01 b 110 Ⓕ
	BIT b, (IX+d)	$Z \leftarrow \overline{(IX+d)}_b$	×	↕	1	×	0	•	4	20	11 011 101 11 001 011 ← d → 01 b 110 Ⓕ
	BIT b, (IY+d)	$Z \leftarrow \overline{(IY+d)}_b$	×	↕	1	×	0	•	4	20	11 111 101 11 001 011 ← d → 01 b 110 Ⓕ
	SET b, r	$r_b \leftarrow 1$	•	•	•	•	•	•	2	8	11 001 011 11 b r ⒺⒻ
	SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	•	•	•	•	2	15	11 001 011 11 b 110 Ⓕ
	SET b, (IX+d)	$(IX+d)_b \leftarrow 1$	•	•	•	•	•	•	4	23	11 011 101 11 001 011 ← d → 11 b 110 Ⓕ
	SET b, (IY+d)	$(IY+d)_b \leftarrow 1$	•	•	•	•	•	•	4	23	11 111 101 11 001 011 ← d → 11 b 110 Ⓕ
	RES b, r	$r_b \leftarrow 0$	•	•	•	•	•	•	2	8	11 001 011 10 b r ⒺⒻ
	RES b, (HL)	$(HL)_b \leftarrow 0$	•	•	•	•	•	•	2	15	11 001 011 10 b 110 Ⓕ
	RES b, (IX+d)	$(IX+d)_b \leftarrow 0$	•	•	•	•	•	•	4	23	11 011 101 11 001 011 ← d → 10 b 110 Ⓕ
	RES b, (IY+d)	$(IY+d)_b \leftarrow 0$	•	•	•	•	•	•	4	23	11 111 101 11 001 011 ← d → 10 b 110 Ⓕ

命令群	ニーモニック	オペレーション	フラグ S Z H P/V N C	バイト	ステート	OPコード 76 543 210
ジャンプ・コール・リターン命令	JP nn	PC←nn	• • • • • • •	3	10	11 000 011 ← n → ← n →
	JP cc, nn	If cc is true PC←nn Otherwise continue	• • • • • • •	3	10	11 cc 010 Ⓖ ← n → ← n →
	JR e	PC←PC+e	• • • • • • •	2	12	00 011 000 ← e−2 →
	JR C, e	If C=0 continue If C=1 PC←PC+e	• • • • • • •	2	7 if C=0 12 if C=1	00 111 000 ← e−2 →
	JR NC, e	If C=1 continue If C=0 PC←PC+e	• • • • • • •	2	7 if C=1 12 if C=0	00 110 000 ← e−2 →
	JR Z, e	If Z=0 continue If Z=1 PC←PC+e	• • • • • • •	2	7 if Z=0 12 if Z=1	00 101 000 ← e−2 →
	JR NZ, e	If Z=1 continue If Z=0 PC←PC+e	• • • • • • •	2	7 if Z=1 12 if Z=0	00 100 000 ← e−2 →
	JP (HL)	PC←HL	• • • • • • •	1	4	11 101 001
	JP (IX)	PC←IX	• • • • • • •	2	8	11 011 101 11 101 001
	JP (IY)	PC←IY	• • • • • • •	2	8	11 111 101 11 101 001
	DJNZ e	B←B−1 if B=0 continue if B≠0 PC←PC+e	• • • • • • •	2	8 if B=0 13 if B≠0	00 010 000 ← e−2 →
	CALL nn	(SP−1)←PC_H (SP−2)←PC_L PC←nn	• • • • • • •	3	17	11 001 101 ← n → ← n →
	CALL cc, nn	If cc is false continue otherwise same as CALL nn	• • • • • • •	3	10 if cc is false 17 if cc is true	11 cc 100 Ⓖ ← n → ← n →
	RET	PC_L←(SP) PC_H←(SP+1)	• • • • • • •	1	10	11 001 001
	RET cc	If cc is false continue otherwise same as RET	• • • • • • •	1	5 if cc is false 11 if cc is true	11 cc 000 Ⓖ
	RETI	Return from interrupt	• • • • • • •	2	14	11 101 101 01 001 101
	RETN	Return from non maskable interrupt	• • • • • • •	2	14	11 101 101 01 000 101
	RST p	(SP−1)←PC_H (SP−2)←PC_L PC_H←0 PC_L←p	• • • • • • •	1	11	11 t 111 Ⓗ
入出力命令	IN A, n	A←(n) $A_{0\sim7}$←n $A_{8\sim15}$←A	• • • • • • •	2	11	11 011 011 ← n →
	IN r, (C)	r←(C) if r=110 only the flags will be affected $A_{0\sim7}$←C $A_{8\sim15}$←B	↕ ↕ ↕ P 0 •	2	12	11 101 101 01 r 000 Ⓔ
	INI	(HL)←(C) B←B−1 HL←HL+1 $A_{0\sim7}$←C $A_{8\sim15}$←B	× ↕③ × × 1 •	2	16	11 101 101 10 100 010
	INIR	(HL)←(C) B←B−1 HL←HL+1 until B=0 $A_{0\sim7}$←C $A_{8\sim15}$←B	× 1 × × 1 •	2	21 if B≠0 16 if B=0	11 101 101 10 110 010
	IND	(HL)←(C) B←B−1 HL←HL−1 $A_{0\sim7}$←C $A_{8\sim15}$←B	× ↕③ × × 1 •	2	16	11 101 101 10 101 010

命令群	ニーモニック	オペレーション	フラグ S Z H P/V N C	バイト	ステート	OPコード 76 543 210
入出力命令	INDR	(HL)←(C) B←B−1 HL←HL−1 until B=0 $A_{0\sim7}$←C $A_{8\sim15}$←B	× 1 × × 1 •	2	21 if B≠0 16 if B=0	11 101 101 10 111 010
	OUT n, A	(n)←A $A_{0\sim7}$←n $A_{8\sim15}$←A	• • • • • •	2	11	11 010 011 ← n →
	OUT (C), r	(C)←r $A_{0\sim7}$←C $A_{8\sim15}$←B	• • • • • •	2	12	11 101 101 01 r 001 Ⓔ
	OUTI	(C)←(HL) B←B−1 HL←HL+1 $A_{0\sim7}$←C $A_{8\sim15}$←B	× ③↕ × × 1 •	2	16	11 101 101 10 100 011
	OTIR	(C)←(HL) B←B−1 HL←HL+1 until B=0 $A_{0\sim7}$←C $A_{8\sim15}$←B	× 1 × × 1 •	2	21 if B≠0 16 if B=0	11 101 101 10 110 011
	OUTD	(C)←(HL) B←B−1 HL←HL−1 $A_{0\sim7}$←C $A_{8\sim15}$←B	× ③↕ × × 1 •	2	16	11 101 101 10 101 011
	OTDR	(C)←(HL) B←B−1 HL←HL−1 until B=0 $A_{0\sim7}$←C $A_{8\sim15}$←B	× 1 × × 1 •	2	21 if B≠0 16 if B=0	11 101 101 10 111 011

Ⓐ		Ⓑ		Ⓒ		Ⓓ		Ⓔ		Ⓕ		Ⓖ				Ⓗ	
Reg	ss dd	Reg	qq	Reg	pp	Reg	rr	Reg	r,r′	Bit	b	cc	Condition		Flag	P	t
BC	00	BC	00	BC	00	BC	00	B	000	0	000	000	NZ	Non Zero	Z	00H	000
DE	01	DE	01	DE	01	DE	01	C	001	1	001	001	Z	Zero	Z	08H	001
HL	10	HL	10	IX	10	IY	10	D	010	2	010	010	NC	Non Carry	C	10H	010
SP	11	AF	11	SP	11	SP	11	E	011	3	011	011	C	Carry	C	18H	011
								H	100	4	100	100	PO	Parity Odd	P/V	20H	100
								L	101	5	101	101	PE	Parity Even	P/V	28H	101
								A	111	6	110	110	P	Sign Positive	S	30H	110
										7	111	111	M	Sign Negative	S	38H	111

d, n ：8ビット・イミーディエト・データ
e ：相対アドレシングの変位置
e−2：eの実効変位置

フラグ

•：影響受けない　↕：演算結果に従った影響を受ける　① BC−1=0 ならば P/V=0, その他 P/V=1
0：リセット　P："1" 偶数パリティ，"0" 奇数パリティ　② A=(HL) ならば Z=1, その他 Z=0
1：セット　V："1"オーバフロー有り，"0"オーバフロー無し　③ B−1=0 ならば Z=1, その他 Z=0
×：不定　IFF：P/Vフラグ← (IFF)

Z80 マシン語←→ニーモニック対応表

機械語 ——→ ニーモニック											
00	NOP		40	LD	B,B	80	ADD	A,B	C0	RET	NZ
01	LD	BC,nn	41	LD	B,C	81	ADD	A,C	C1	POP	BC
02	LD	(BC),A	42	LD	B,D	82	ADD	A,D	C2	JP	NZ,nn
03	INC	BC	43	LD	B,E	83	ADD	A,E	C3	JP	nn
04	INC	B	44	LD	B,H	84	ADD	A,H	C4	CALL	NZ,nn
05	DEC	B	45	LD	B,L	85	ADD	A,L	C5	PUSH	BC
06	LD	B,n	46	LD	B,(HL)	86	ADD	A,(HL)	C6	ADD	A,n
07	RLCA		47	LD	B,A	87	ADD	A,A	C7	RST	00H
08	EX	AF,AF	48	LD	C,B	88	ADC	A,B	C8	RET	Z
09	ADD	HL,BC	49	LD	C,C	89	ADC	A,C	C9	RET	
0A	LD	A,(BC)	4A	LD	C,D	8A	ADC	A,D	CA	JP	Z,nn
0B	DEC	BC	4B	LD	C,E	8B	ADC	A,E	CB		
0C	INC	C	4C	LD	C,H	8C	ADC	A,H	CC	CALL	Z,nn
0D	DEC	C	4D	LD	C,L	8D	ADC	A,L	CD	CALL	nn
0E	LD	C,n	4E	LD	C,(HL)	8E	ADC	A,(HL)	CE	ADC	A,n
0F	RRCA		4F	LD	C,A	8F	ADC	A,A	CF	RST	08H
10	DJNZ	e	50	LD	D,B	90	SUB	B	D0	RET	NC
11	LD	DE,nn	51	LD	D,C	91	SUB	C	D1	POP	DE
12	LD	(DE),A	52	LD	D,D	92	SUB	D	D2	JP	NC,nn
13	INC	DE	53	LD	D,E	93	SUB	E	D3	OUT	n,A
14	INC	D	54	LD	D,H	94	SUB	H	D4	CALL	NC,nn
15	DEC	D	55	LD	D,L	95	SUB	L	D5	PUSH	DE
16	LD	D,n	56	LD	D,(HL)	96	SUB	(HL)	D6	SUB	n
17	RLA		57	LD	D,A	97	SUB	A	D7	RST	10H
18	JR	e	58	LD	E,B	98	SBC	A,B	D8	RET	C
19	ADD	HL,DE	59	LD	E,C	99	SBC	A,C	D9	EXX	
1A	LD	A,(DE)	5A	LD	E,D	9A	SBC	A,D	DA	JP	C,nn
1B	DEC	DE	5B	LD	E,E	9B	SBC	A,E	DB	IN	A,n
1C	INC	E	5C	LD	E,H	9C	SBC	A,H	DC	CALL	C,nn
1D	DEC	E	5D	LD	E,L	9D	SBC	A,L	DD		
1E	LD	E,n	5E	LD	E,(HL)	9E	SBC	A,(HL)	DE	SBC	A,n
1F	RRA		5F	LD	E,A	9F	SBC	A,A	DF	RST	18H
20	JR	NZ,e	60	LD	H,B	A0	AND	B	E0	RET	PO
21	LD	HL,nn	61	LD	H,C	A1	AND	C	E1	POP	HL
22	LD	(nn),HL	62	LD	H,D	A2	AND	D	E2	JP	PO,nn
23	INC	HL	63	LD	H,E	A3	AND	E	E3	EX	(SP),HL
24	INC	H	64	LD	H,H	A4	AND	H	E4	CALL	PO,nn
25	DEC	H	65	LD	H,L	A5	AND	L	E5	PUSH	HL
26	LD	H,n	66	LD	H,(HL)	A6	AND	(HL)	E6	AND	n
27	DAA		67	LD	H,A	A7	AND	A	E7	RST	20H
28	JR	Z,e	68	LD	L,B	A8	XOR	B	E8	RET	PE
29	ADD	HL,HL	69	LD	L,C	A9	XOR	C	E9	JP	(HL)
2A	LD	HL,(nn)	6A	LD	L,D	AA	XOR	D	EA	JP	PE,nn
2B	DEC	HL	6B	LD	L,E	AB	XOR	E	EB	EX	DE,HL
2C	INC	L	6C	LD	L,H	AC	XOR	H	EC	CALL	PE,nn
2D	DEC	L	6D	LD	L,L	AD	XOR	L	ED		
2E	LD	L,n	6E	LD	L,(HL)	AE	XOR	(HL)	EE	XOR	n
2F	CPL		6F	LD	L,A	AF	XOR	A	EF	RST	28H
30	JR	NC,e	70	LD	(HL),B	B0	OR	B	F0	RET	P
31	LD	SP,nn	71	LD	(HL),C	B1	OR	C	F1	POP	AF
32	LD	(nn),A	72	LD	(HL),D	B2	OR	D	F2	JP	P,nn
33	INC	SP	73	LD	(HL),E	B3	OR	E	F3	DI	
34	INC	(HL)	74	LD	(HL),H	B4	OR	H	F4	CALL	P,nn
35	DEC	(HL)	75	LD	(HL),L	B5	OR	L	F5	PUSH	AF
36	LD	(HL),n	76	HALT		B6	OR	(HL)	F6	OR	n
37	SCF		77	LD	(HL),A	B7	OR	A	F7	RST	30H
38	JR	C,e	78	LD	A,B	B8	CP	B	F8	RET	M
39	ADD	HL,SP	79	LD	A,C	B9	CP	C	F9	LD	SP,HL
3A	LD	A,(nn)	7A	LD	A,D	BA	CP	D	FA	JP	M,nn
3B	DEC	SP	7B	LD	A,E	BB	CP	E	FB	EI	
3C	INC	A	7C	LD	A,H	BC	CP	H	FC	CALL	M,nn
3D	DEC	A	7D	LD	A,L	BD	CP	L	FD		
3E	LD	A,n	7E	LD	A,(HL)	BE	CP	(HL)	FE	CP	n
3F	CCF		7F	LD	A,A	BF	CP	A	FF	RST	38H

CB ××											
00	RLC	B	40	BIT	0,B	80	RES	0,B	C0	SET	0,B
01	RLC	C	41	BIT	0,C	81	RES	0,C	C1	SET	0,C
02	RLC	D	42	BIT	0,D	82	RES	0,D	C2	SET	0,D
03	RLC	E	43	BIT	0,E	83	RES	0,E	C3	SET	0,E
04	RLC	H	44	BIT	0,H	84	RES	0,H	C4	SET	0,H
05	RLC	L	45	BIT	0,L	85	RES	0,L	C5	SET	0,L
06	RLC	(HL)	46	BIT	0,(HL)	86	RES	0,(HL)	C6	SET	0,(HL)
07	RLC	A	47	BIT	0,A	87	RES	0,A	C7	SET	0,A
08	RRC	B	48	BIT	1,B	88	RES	1,B	C8	SET	1,B
09	RRC	C	49	BIT	1,C	89	RES	1,C	C9	SET	1,C
0A	RRC	D	4A	BIT	1,D	8A	RES	1,D	CA	SET	1,D
0B	RRC	E	4B	BIT	1,E	8B	RES	1,E	CB	SET	1,E
0C	RRC	H	4C	BIT	1,H	8C	RES	1,H	CC	SET	1,H
0D	RRC	L	4D	BIT	1,L	8D	RES	1,L	CD	SET	1,L
0E	RRC	(HL)	4E	BIT	1,(HL)	8E	RES	1,(HL)	CE	SET	1,(HL)
0F	RRC	A	4F	BIT	1,A	8F	RES	1,A	CF	SET	1,A
10	RL	B	50	BIT	2,B	90	RES	2,B	D0	SET	2,B
11	RL	C	51	BIT	2,C	91	RES	2,C	D1	SET	2,C
12	RL	D	52	BIT	2,D	92	RES	2,D	D2	SET	2,D
13	RL	E	53	BIT	2,E	93	RES	2,E	D3	SET	2,E
14	RL	H	54	BIT	2,H	94	RES	2,H	D4	SET	2,H
15	RL	L	55	BIT	2,L	95	RES	2,L	D5	SET	2,L
16	RL	(HL)	56	BIT	2,(HL)	96	RES	2,(HL)	D6	SET	2,(HL)
17	RL	A	57	BIT	2,A	97	RES	2,A	D7	SET	2,A
18	RR	B	58	BIT	3,B	98	RES	3,B	D8	SET	3,B
19	RR	C	59	BIT	3,C	99	RES	3,C	D9	SET	3,C
1A	RR	D	5A	BIT	3,D	9A	RES	3,D	DA	SET	3,D
1B	RR	E	5B	BIT	3,E	9B	RES	3,E	DB	SET	3,E
1C	RR	H	5C	BIT	3,H	9C	RES	3,H	DC	SET	3,H
1D	RR	L	5D	BIT	3,L	9D	RES	3,L	DD	SET	3,L
1E	RR	(HL)	5E	BIT	3,(HL)	9E	RES	3,(HL)	DE	SET	3,(HL)
1F	RR	A	5F	BIT	3,A	9F	RES	3,A	DF	SET	3,A
20	SLA	B	60	BIT	4,B	A0	RES	4,B	E0	SET	4,B
21	SLA	C	61	BIT	4,C	A1	RES	4,C	E1	SET	4,C
22	SLA	D	62	BIT	4,D	A2	RES	4,D	E2	SET	4,D
23	SLA	E	63	BIT	4,E	A3	RES	4,E	E3	SET	4,E
24	SLA	H	64	BIT	4,H	A4	RES	4,H	E4	SET	4,H
25	SLA	L	65	BIT	4,L	A5	RES	4,L	E5	SET	4,L
26	SLA	(HL)	66	BIT	4,(HL)	A6	RES	4,(HL)	E6	SET	4,(HL)
27	SLA	A	67	BIT	4,A	A7	RES	4,A	E7	SET	4,A
28	SRA	B	68	BIT	5,B	A8	RES	5,B	E8	SET	5,B
29	SRA	C	69	BIT	5,C	A9	RES	5,C	E9	SET	5,C
2A	SRA	D	6A	BIT	5,D	AA	RES	5,D	EA	SET	5,D
2B	SRA	E	6B	BIT	5,E	AB	RES	5,E	EB	SET	5,E
2C	SRA	H	6C	BIT	5,H	AC	RES	5,H	EC	SET	5,H
2D	SRA	L	6D	BIT	5,L	AD	RES	5,L	ED	SET	5,L
2E	SRA	(HL)	6E	BIT	5,(HL)	AE	RES	5,(HL)	EE	SET	5,(HL)
2F	SRA	A	6F	BIT	5,A	AF	RES	5,A	EF	SET	5,A
30			70	BIT	6,B	B0	RES	6,B	F0	SET	6,B
31			71	BIT	6,C	B1	RES	6,C	F1	SET	6,C
32			72	BIT	6,D	B2	RES	6,D	F2	SET	6,D
33			73	BIT	6,E	B3	RES	6,E	F3	SET	6,E
34			74	BIT	6,H	B4	RES	6,H	F4	SET	6,H
35			75	BIT	6,L	B5	RES	6,L	F5	SET	6,L
36			76	BIT	6,(HL)	B6	RES	6,(HL)	F6	SET	6,(HL)
37			77	BIT	6,A	B7	RES	6,A	F7	SET	6,A
38	SRL	B	78	BIT	7,B	B8	RES	7,B	F8	SET	7,B
39	SRL	C	79	BIT	7,C	B9	RES	7,C	F9	SET	7,C
3A	SRL	D	7A	BIT	7,D	BA	RES	7,D	FA	SET	7,D
3B	SRL	E	7B	BIT	7,E	BB	RES	7,E	FB	SET	7,E
3C	SRL	H	7C	BIT	7,H	BC	RES	7,H	FC	SET	7,H
3D	SRL	L	7D	BIT	7,L	BD	RES	7,L	FD	SET	7,L
3E	SRL	(HL)	7E	BIT	7,(HL)	BE	RES	7,(HL)	FE	SET	7,(HL)
3F	SRL	A	7F	BIT	7,A	BF	RES	7,A	FF	SET	7,A

DD ××					ED ××			FD ××				
09			ADD	IX, BC	40	IN	B, (C)	09			ADD	IY, BC
19			ADD	IX, DE	41	OUT	(C), B	19			ADD	IY, DE
21			LD	IX, nn	42	SBC	HL, BC	21			LD	IY, nn
22			LD	(nn), IX	43	LD	(nn), BC	22			LD	(nn), IY
23			INC	IX	44	NEG		23			INC	IY
29			ADD	IX, IX	45	RETN		29			ADD	IY, IY
2A			LD	IX, (nn)	46	IM	0	2A			LD	IY, (nn)
2B			DEC	IX	47	LD	I, A	2B			DEC	IY
34			INC	(IX+d)	48	IN	C, (C)	34			INC	(IY+d)
35			DEC	(IX+d)	49	OUT	(C), C	35			DEC	(IY+d)
36			LD	(IX+d), n	4A	ADC	HL, BC	36			LD	(IY+d), n
39			ADD	IX, SP	4B	LD	BC, (nn)	39			ADD	IY, SP
46			LD	B, (IX+d)	4D	RETI		46			LD	B, (IY+d)
4E			LD	C, (IX+d)	4F	LD	R, A	4E			LD	C, (IY+d)
56			LD	D, (IX+d)	50	IN	D, (C)	56			LD	D, (IY+d)
5E			LD	E, (IX+d)	51	OUT	(C), D	5E			LD	E, (IY+d)
66			LD	H, (IX+d)	52	SBC	HL, DE	66			LD	H, (IY+d)
6E			LD	L, (IX+d)	53	LD	(nn), DE	6E			LD	L, (IY+d)
70			LD	(IX+d), B	56	IM	1	70			LD	(IY+d), B
71			LD	(IX+d), C	57	LD	A, I	71			LD	(IY+d), C
72			LD	(IX+d), D	58	IN	E, (C)	72			LD	(IY+d), D
73			LD	(IX+d), E	59	OUT	(C), E	73			LD	(IY+d), E
74			LD	(IX+d), H	5A	ADC	HL, DE	74			LD	(IY+d), H
75			LD	(IX+d), L	5B	LD	DE, (nn)	75			LD	(IY+d), L
77			LD	(IX+d), A	5E	IM	2	77			LD	(IY+d), A
7E			LD	A, (IX+d)	5F	LD	A, R	7E			LD	A, (IY+d)
86			ADD	A, (IX+d)	60	IN	H, (C)	86			ADD	A, (IY+d)
8E			ADC	A, (IX+d)	61	OUT	(C), H	8E			ADC	A, (IY+d)
96			SUB	(IX+d)	62	SBC	HL, HL	96			SUB	(IY+d)
9E			SBC	A, (IX+d)	67	RRD		9E			SBC	A, (IY+d)
A6			AND	(IX+d)	68	IN	L, (C)	A6			AND	(IY+d)
AE			XOR	(IX+d)	69	OUT	(C), L	AE			XOR	(IY+d)
B6			OR	(IX+d)	6A	ADC	HL, HL	B6			OR	(IY+d)
BE			CP	(IX+d)	6F	RLD		BE			CP	(IY+d)
CB	d	06	RLC	(IX+d)	72	SBC	HL, SP	CB	d	06	RLC	(IY+d)
CB	d	0E	RRC	(IX+d)	73	LD	(nn), SP	CB	d	0E	RRC	(IY+d)
CB	d	16	RL	(IX+d)	78	IN	A, (C)	CB	d	16	RL	(IY+d)
CB	d	1E	RR	(IX+d)	79	OUT	(C), A	CB	d	1E	RR	(IY+d)
CB	d	26	SLA	(IX+d)	7A	ADC	HL, SP	CB	d	26	SLA	(IY+d)
CB	d	2E	SRA	(IX+d)	7B	LD	SP, (nn)	CB	d	2E	SRA	(IY+d)
CB	d	3E	SRL	(IX+d)	A0	LDI		CB	d	3E	SRL	(IY+d)
CB	d	46	BIT	0, (IX+d)	A1	CPI		CB	d	46	BIT	0, (IY+d)
CB	d	4E	BIT	1, (IX+d)	A2	INI		CB	d	4E	BIT	1, (IY+d)
CB	d	56	BIT	2, (IX+d)	A3	OUTI		CB	d	56	BIT	2, (IY+d)
CB	d	5E	BIT	3, (IX+d)	A8	LDD		CB	d	5E	BIT	3, (IY+d)
CB	d	66	BIT	4, (IX+d)	A9	CPD		CB	d	66	BIT	4, (IY+d)
CB	d	6E	BIT	5, (IX+d)	AA	IND		CB	d	6E	BIT	5, (IY+d)
CB	d	76	BIT	6, (IX+d)	AB	OUTD		CB	d	76	BIT	6, (IY+d)
CB	d	7E	BIT	7, (IX+d)	B0	LDIR		CB	d	7E	BIT	7, (IY+d)
CB	d	86	RES	0, (IX+d)	B1	CPIR		CB	d	86	RES	0, (IY+d)
CB	d	8E	RES	1, (IX+d)	B2	INIR		CB	d	8E	RES	1, (IY+d)
CB	d	96	RES	2, (IX+d)	B3	OTIR		CB	d	96	RES	2, (IY+d)
CB	d	9E	RES	3, (IX+d)	B8	LDDR		CB	d	9E	RES	3, (IY+d)
CB	d	A6	RES	4, (IX+d)	B9	CPDR		CB	d	A6	RES	4, (IY+d)
CB	d	AE	RES	5, (IX+d)	BA	INDR		CB	d	AE	RES	5, (IY+d)
CB	d	B6	RES	6, (IX+d)	BB	OTDR		CB	d	B6	RES	6, (IY+d)
CB	d	BE	RES	7, (IX+d)				CB	d	BE	RES	7, (IY+d)
CB	d	C6	SET	0, (IX+d)				CB	d	C6	SET	0, (IY+d)
CB	d	CE	SET	1, (IX+d)				CB	d	CE	SET	1, (IY+d)
CB	d	D6	SET	2, (IX+d)				CB	d	D6	SET	2, (IY+d)
CB	d	DE	SET	3, (IX+d)				CB	d	DE	SET	3, (IY+d)
CB	d	E6	SET	4, (IX+d)				CB	d	E6	SET	4, (IY+d)
CB	d	EE	SET	5, (IX+d)				CB	d	EE	SET	5, (IY+d)
CB	d	F6	SET	6, (IX+d)				CB	d	F6	SET	6, (IY+d)
CB	d	FE	SET	7, (IX+d)				CB	d	FE	SET	7, (IY+d)
E1			POP	IX				E1			POP	IY
E3			EX	(SP), IX				E3			EX	(SP), IY
E5			PUSH	IX				E5			PUSH	IY
E9			JP	(IX)				E9			JP	(IY)
F9			LD	SP, IX				F9			LD	SP, IY

Z80　ニーモニック←→マシン語対照表

8ビット・ロード

×	I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX+d)	(IY+d)	(nn)	n
LD A, ×	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A n n	3E n
LD B, ×			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 n
LD C, ×			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E n
LD D, ×			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 n
LD E, ×			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		1E n
LD H, ×			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		26 n
LD L, ×			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		2E n
LD (HL), ×			77	70	71	72	73	74	75							36 n
LD (BC), ×			02													
LD (DE), ×			12													
LD (IX+d), ×			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
LD (IY+d), ×			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
LD (nn), ×			32 n n													
LD I, ×			ED 47													
LD R, ×			ED 4F													

16ビット・ロード

×	AF	BC	DE	HL	SP	IX	IY	nn	(nn)
LD AF, ×									
LD BC, ×								01 n n	ED 4B n n
LD DE, ×								11 n n	ED 5B n n
LD HL, ×								21 n n	2A n n
LD SP, ×				F9		DD F9	FD F9	31 n n	ED 7B n n
LD IX, ×								DD 21 n n	DD 2A n n
LD IY, ×								FD 21 n n	FD 2A n n
LD (nn), ×		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n		
PUSH ×	F5	C5	D5	E5		DD E5	FD E5		
POP ×	F1	C1	D1	E1		DD E1	FD E1		

ブロック転送

LDI	ED A0
LDIR	ED B0
LDD	ED A8
LDDR	ED B8

ブロック・サーチ

CPI	ED A1
CPIR	ED B1
CPD	ED A9
CPDR	ED B9

8ビット算術論理演算

×	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
ADD A, ×	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
ADC A, ×	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUB ×	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SBC A, ×	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
AND ×	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
XOR ×	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
OR ×	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
CP ×	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INC ×	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DEC ×	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

CPUコントロール

NOP	00
HALT	76
DI	F3
EI	FB
IM 0	ED 46
IM 1	ED 56
IM 2	ED 5E

16ビット算術演算

×	BC	DE	HL	SP	IX	IY
ADD HL, ×	09	19	29	39		
ADD IX, ×	DD 09	DD 19		DD 39	DD 29	
ADD IY, ×	FD 09	FD 19		FD 39		FD 29
ADC HL, ×	ED 4A	ED 5A	ED 6A	ED 7A		
SBC HL, ×	ED 42	ED 52	ED 62	ED 72		
INC ×	03	13	23	33	DD 23	FD 23
DEC ×	0B	1B	2B	3B	DD 2B	FD 2B

エクスチェンジ

EX AF, AF'	08
EX DE, HL	EB
EX (SP), HL	E3
EX (SP), IX	DD E3
EX (SP), IY	FD E3
EXX	D9

アキュムレータ操作

DAA	27
CPL	2F
NEG	ED 44
CCF	3F
SCF	37

ローテート，シフト

×	A	B	C	D	E	H	L	(HL)	(IX +d)	(IY +d)
RLC ×	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
RRC ×	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
RL ×	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
RR ×	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
SLA ×	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
SRA ×	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
SRL ×	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
RLD								ED 6F		
RRD								ED 67		

	A
RLCA	07
RRCA	0F
RLA	17
RRA	1F

ジャンプ，コール，リターン

×	UN COND	C	NC	Z	NZ	PE	PO	M	P	
JP ×, nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JR ×, e	18 e−2	38 e−2	30 e−2	28 e−2	20 e−2					
JP (HL)	E9									
JP (IX)	DD E9									
JP (IY)	FD E9									
CALL ×, nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
DJNZ e										10 e−2
RET ×	C9	D8	D0	C8	C0	E8	E0	F8	F0	
RETI	ED 4D									
RETN	ED 45									

リスタート

RST 00H	C7
RST 08H	CF
RST 10H	D7
RST 18H	DF
RST 20H	E7
RST 28H	EF
RST 30H	F7
RST 38H	FF

ビット操作

×	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
BIT 0, ×	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
BIT 1, ×	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
BIT 2, ×	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
BIT 3, ×	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
BIT 4, ×	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
BIT 5, ×	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
BIT 6, ×	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76	DD CB d 76	FD CB d 76
BIT 7, ×	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
RES 0, ×	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
RES 1, ×	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
RES 2, ×	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
RES 3, ×	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E
RES 4, ×	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
RES 5, ×	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
RES 6, ×	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
RES 7, ×	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
SET 0, ×	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
SET 1, ×	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
SET 2, ×	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6
SET 3, ×	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
SET 4, ×	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
SET 5, ×	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
SET 6, ×	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
SET 7, ×	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE

入　力

IN A, n	DB n
IN A, (C)	ED 78
IN B, (C)	ED 40
IN C, (C)	ED 48
IN D, (C)	ED 50
IN E, (C)	ED 58
IN H, (C)	ED 60
IN L, (C)	ED 68
INI	ED A2
INIR	ED B2
IND	ED AA
INDR	ED BA

出　力

OUT n, A	D3 n
OUT (C), A	ED 79
OUT (C), B	ED 41
OUT (C), C	ED 49
OUT (C), D	ED 51
OUT (C), E	ED 59
OUT (C), H	ED 61
OUT (C), L	ED 69
OUTI	ED A3
OTIR	ED B3
OUTD	ED AB
OTDR	ED BB

あとがき

本書はZ80（μPD780）のインストラクションそのものの入門書ではありません．マシン語の基礎的な勉強をされたい方は下記の参考文献をご参照ください．

μCOM-82ユーザーズマニュアル	NEC日本電気株式会社
μCOM-82インストラクション活用表	NEC日本電気株式会社
マイクロコンピュータZ80ハンドブック〔I〕	シャープ株式会社
Z80マイコンプログラムテクニック	電波新聞社

本書は独自に解析した結果を出版したものであり，書き足らぬ点や，至らぬ点も多々あろうかと存じます．読者諸兄の御叱正を賜れば幸です．

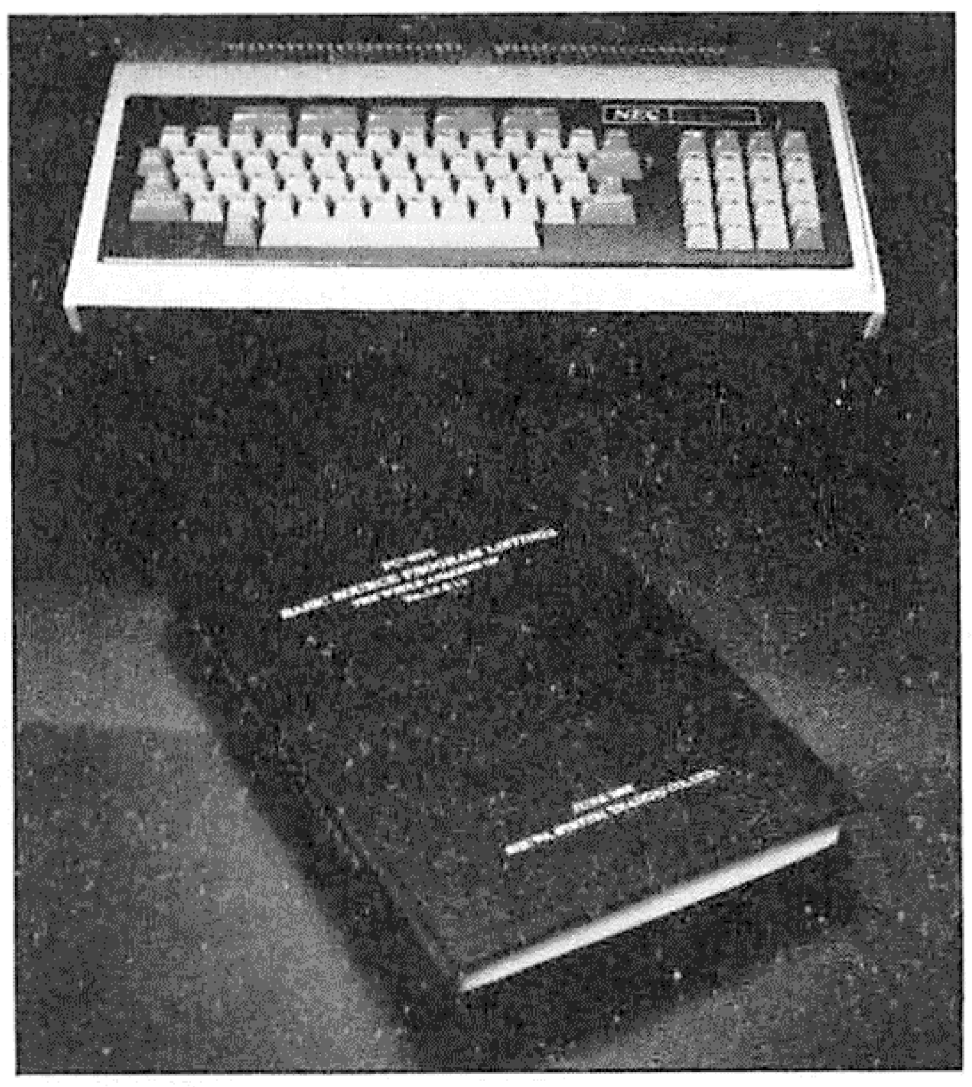
なお，ご意見，ご質問がございましたら出版元まで，**書面**にてお寄せくださいますようお願いいたします．

本書の内容に関して，運用上の影響については責任を負いかねますのでご了承下さい．

著者代表　川村　清
発 行 者　牧谷秀昭
発 行 所　秀和システムトレーディング株式会社
〒106 東京都港区西麻布1-4-20
印 刷 所　東京スガキ印刷株式会社

落丁，乱丁はお取替えします。　**定価 2,800円**

PC-8000 システムアップ・シリーズ

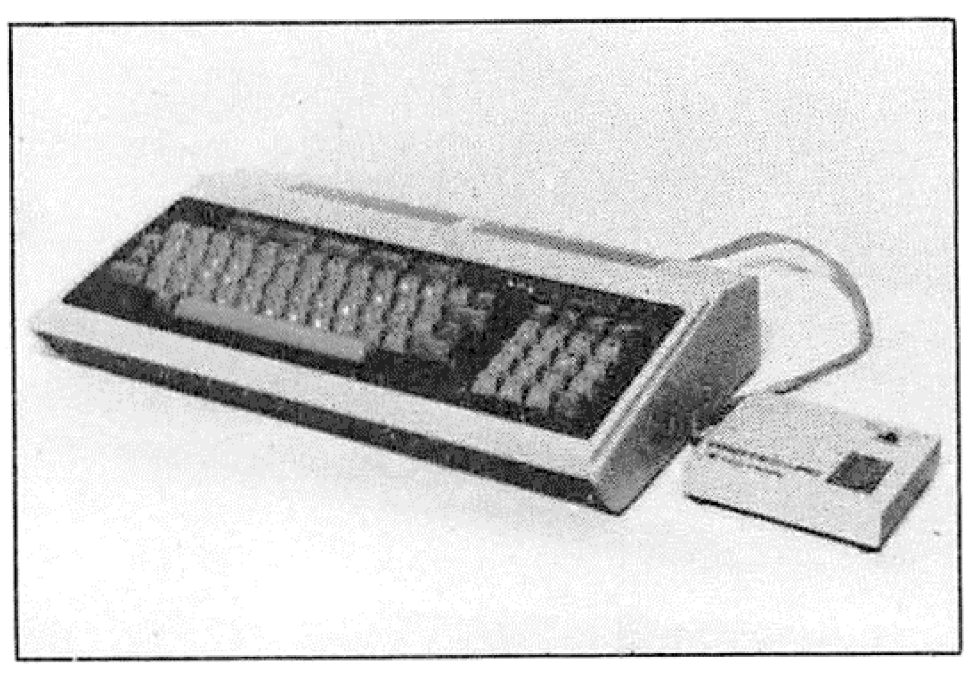


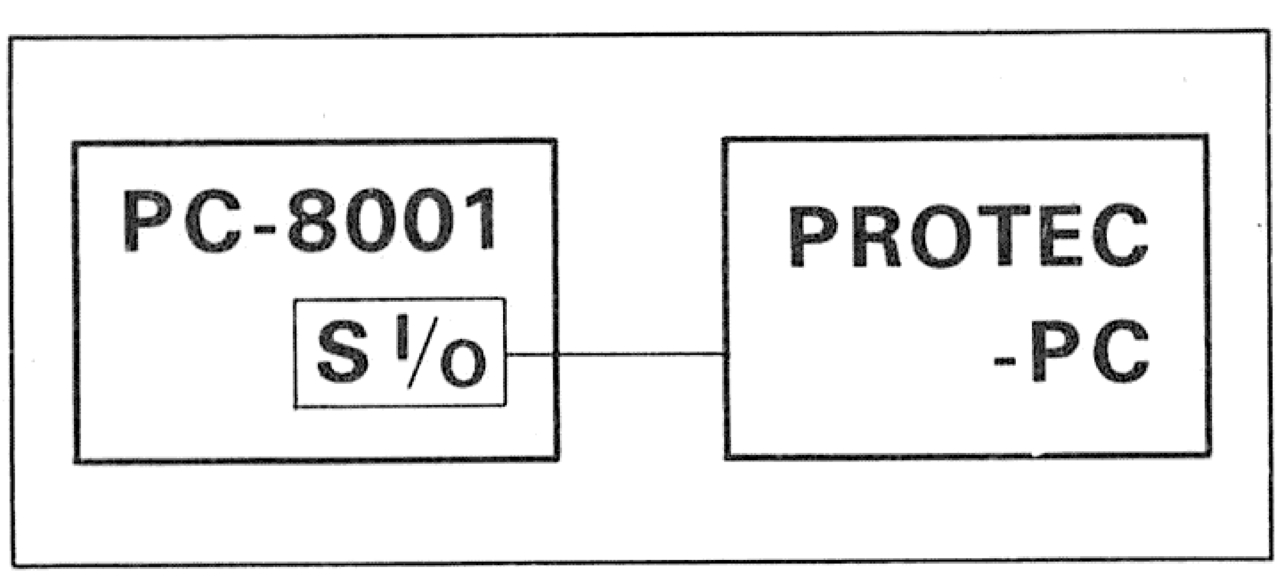
（B5判　上製　全350ページ）

■PC-8001 BASIC SOURCE PROGRAM LISTINGS ¥9,800

N-BASIC　ソース　プログラムリスト

- 独自に解析した、N-BASICインタプリタの内容を一挙に公開!!
- 内部サブルーチンをさらに詳しく研究、活用するための、必読書。
- 『マシン語活用ハンドブック』の理解を助けます。
- 最高の実績を誇る、N-BASICの真髄に迫った本書は、名実ともにパソコンユーザーの“宝„です。



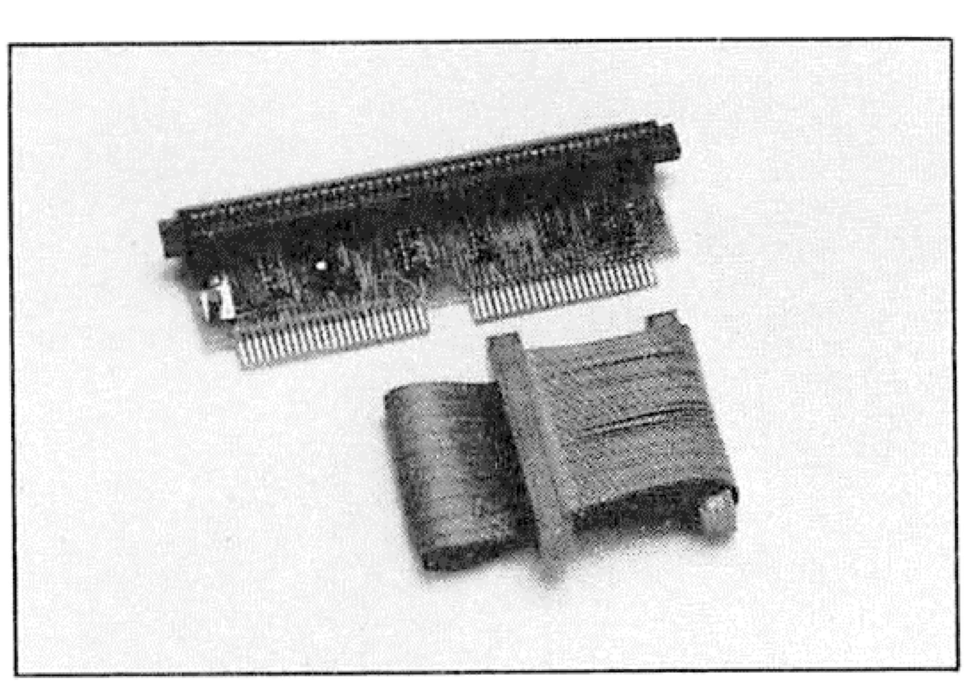


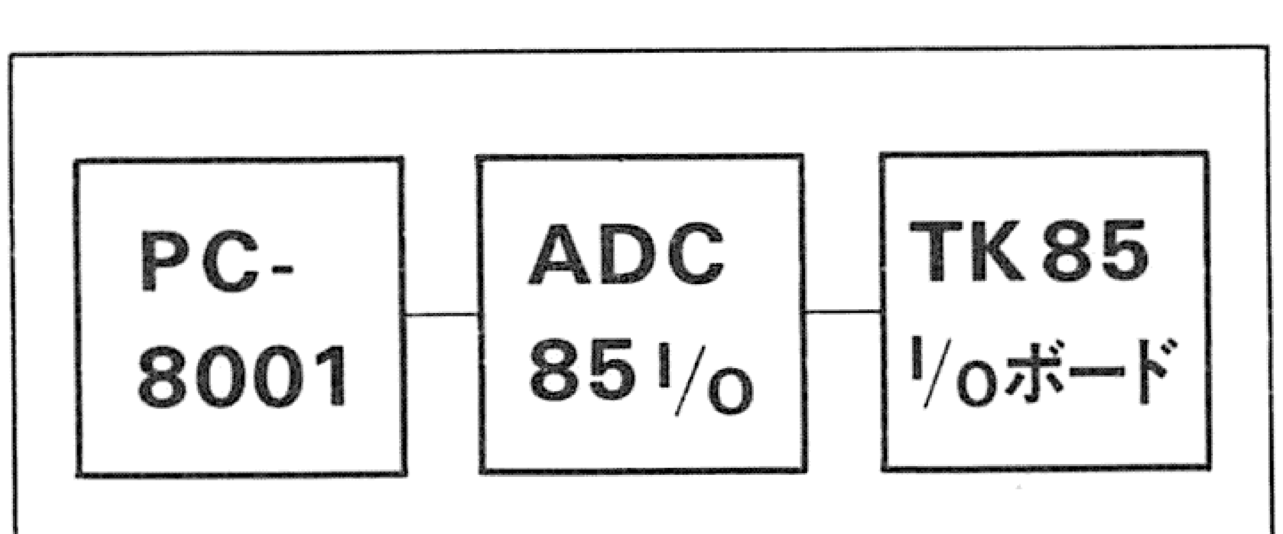
（接続図）

■PROTEC-PC　¥29,800

PC-8001用EP-ROMライター

- 2716、2732、2732Aに対応。
- 特別な外部電源を要しません。
- PC−8001のSIOソケットに接続するのみで動作可能です。
- 転送スピード　4,800ボー
- NEC4ビット、1チップCPU搭載のため動作はきわめて安定しています。
- オペレーションプログラムはカセットテープにて提供されており、BASICを用いて簡単な会話形式で操作できます。





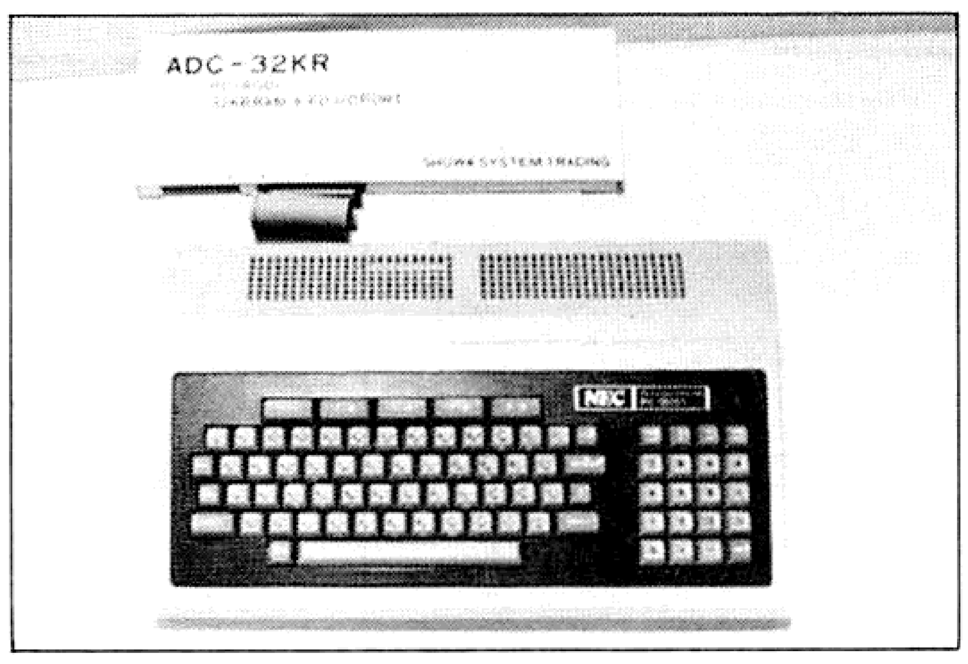
（接続図）

■ADC-85 I/O　¥19,800

PC-8001用 I/Oアダプター

- TK85 I/OボードがPC−8001で使えます。
- ADコンバータ、DAコンバータ、デイジタル I/Oポート、サウンドジェネレータなどのコントロール機能をPC−8001で容易に発揮できます。BASIC言語でTK85 I/Oボードを制御できるため、簡単に取り扱えます。
- TK85 I/Oボードを用いて、AD、DAなどの簡単な実験、制御、研究が行えます。
- 振張ユニット(PC−8011)を介しての接続も可能です。

（TK85 I/OボードはJMCの製品です）

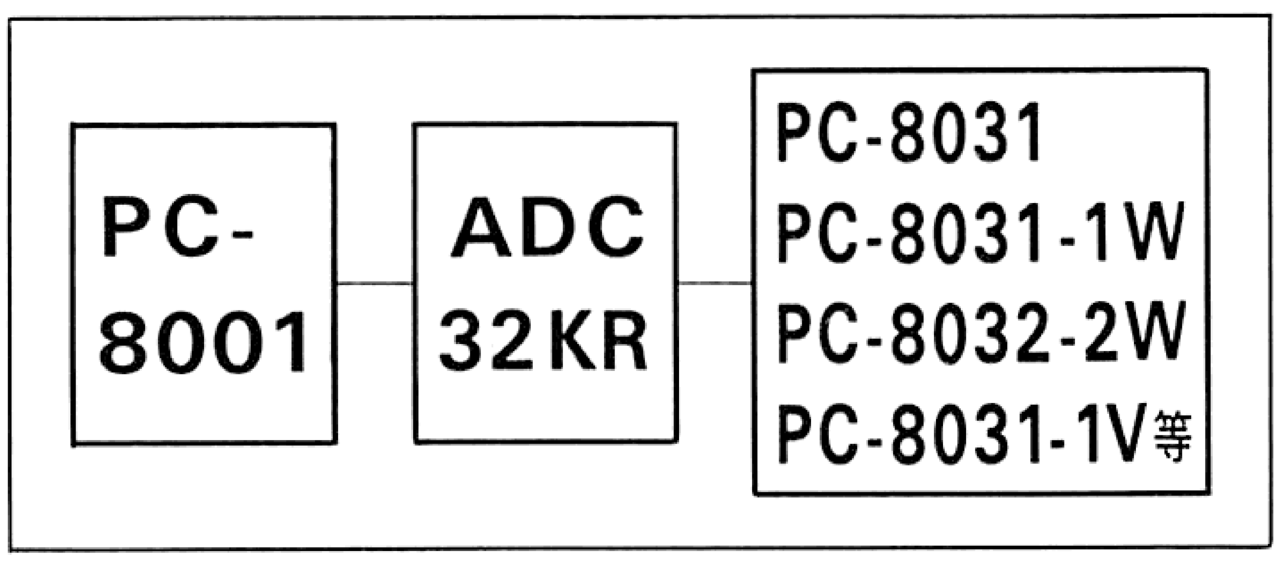


■ADC-32KR　¥48,000

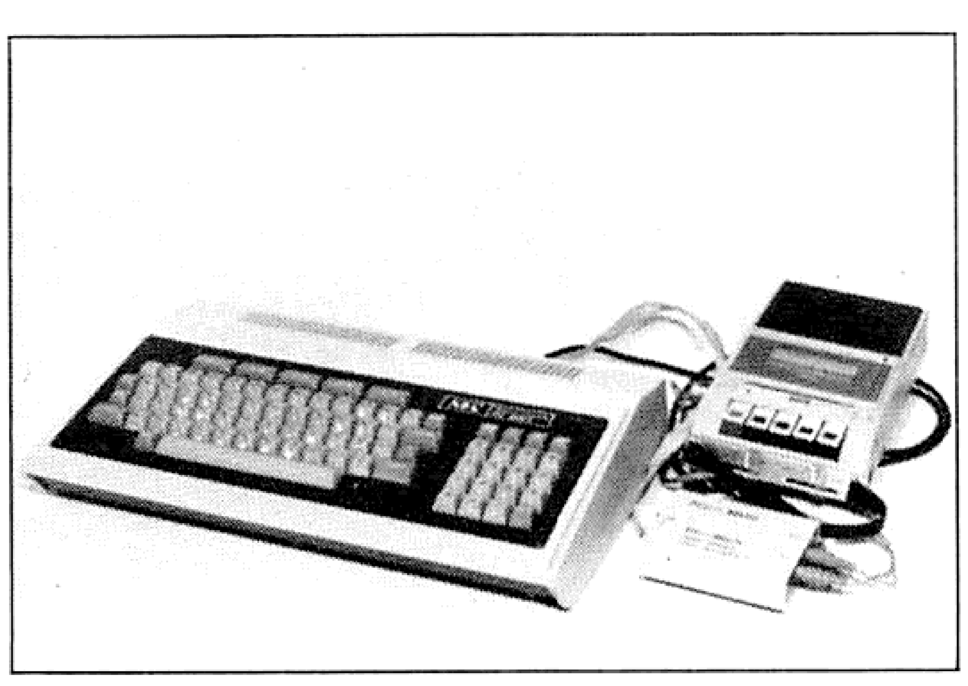
PC-8001用増設RAMボード I/Oポート付

- 拡張ユニット（PC−8011）がなくても、64KRAMシステムが実現します。
- ミニディスク用I/Oポート内蔵。
- このボードのみで、CP/M(V2.2)、NEC CP/M(PCS−101−1W)、PASCAL、NECDOSを走らせる事が出来ます。
- PC−8012、PC−8013の32K RAMボードとして、マザーバスに挿入して使用することも可能です。
- 上記の場合ミニディスク用I/Oポートは汎用I/Oポートとして開放されます。
- N-BASICのユーザーズエリアを拡張する事も出来ます。（約 8K）
- N-BASIC ROM の内容を本RAMボードに転送しプログラムを任意に変更、利用することが可能になります。

（CP/Mはデジタルリサーチ社の登録商標）



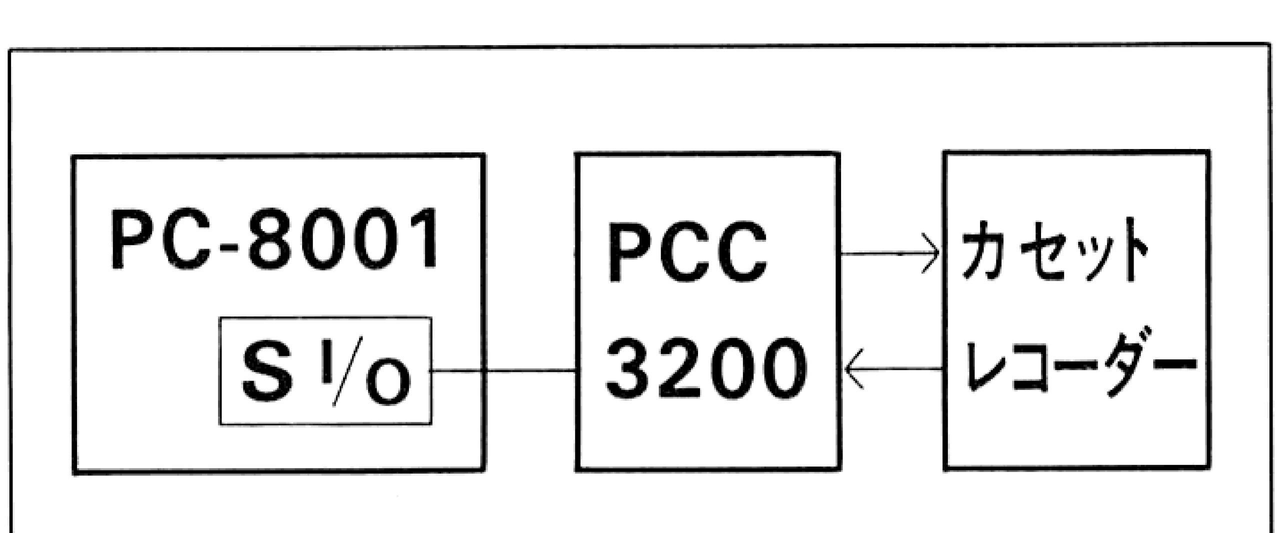
（接続図）



■PCC-3200　¥19,800

PC-8001用高速カセットインターフェース

- 特別な外部電源は不用です。
- NEC4ビット、1チップCPU搭載のため動作はきわめて安定しています。
- プログラムのアペンド機能をもっています。
- コントロールプログラムのカセットテープ及びROM付
- カセットテープのロード セーブ 時間が1/5に短縮されます。（600ボー対比）



（接続図）

対応機能			
	CLOAD	APPEND	L
	CSAVE	RENEW	LV
	CLOAD?	PRINT#-1	W
		INPUT#-1	

●全国有名マイコンショップにて販売中!!

製造・発売元

秀和システムトレーディング株式会社

〒106　東京都港区西麻布1−4−20

TEL (03) 470−4941

SHUWA SYSTEM TRADING CO.,LTD.

定価　2,800円